VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Měření polohy mobilního robotu v budově

Measuring the Position of a Mobile Robot in the Building

Michal Jaworski

Bakalářská práce

Vedoucí práce: Ing. David Vala

Ostrava, Rok: 2021

Abstrakt:

Úkolem bakalářské práce bylo navrhnout systém, který by měl být schopen měřit polohu mobilního robota v budově. Bakalářská práce řeší návrh měření polohy mobilního robota a mapováním okolí robota. Tato práce je rozdělaná na dvě části. První část je teoretická, zahrnuje metody lokalizace a možné senzory pro měření. Druhá část je zaměřena na použité komponenty a realizaci funkčnosti.

Klíčová slova:

ROS, SLAM, Lidar, ultrazvukový měřič vzdálenosti, robot, Arduino.

Abstract:

The task of the bachelor's thesis was to design a system that should be able to measure the mobile robot position a in a building. The bachelor's thesis deals with the design of measuring the position of a mobile robot and robot's surroundings mapping. This thesis is divided into two parts. The first part is theoretical, includes localization methods and possible sensors for measurement. The second part is focused on the components used and functionality implementation.

Keywords:

ROS, SLAM, Lidar, Ultrasonic sensor, robot, Arduino.

Obsah:

Seznam zkratek a symbolů **5**

Seznam obrázků **6**

Seznam tabulek **8**

1 Úvod **9**

2 Lokalizace **10**

2.1 Absolutní lokalizace 10

2.1.1. Triangulace 10

2.1.2. Trilaterace 10

2.1.3. 2D trilaterace 11

2.1.4. 3D trilaterace 11

2.2 Relativní lokalizace 12

2.2.1. Dead Reckoning 12

2.2.2. Odometrie 13

2.2.3. Inerciální určování polohy 14

2.2.4. INS s pevnou orientací akcelerometrů v prostoru 15

2.2.5. INS s akcelerometry pevně spojitými se zařízením 15

2.3 ROS 15

3 SLAM **16**

3.1 Landmark extraction 17

3.1.1. Spikes algoritmus 17

3.1.2. Sliding window algoritmus 17

3.2 Data association 17

3.3 SLAM koncept 18

3.3.1. EKF SLAM 18

3.3.2. FastSLAM 18

4 Snímače pro lokalizaci **18**

4.1 Optický senzor 18

4.1.1. IR senzor 21

4.1.2. Laserový senzor 21

4.2 Ultrazvukový senzor vzdálenosti 21

4.3 Radar neboli radiolokátor 22

4.4 Detektor barvy 23

4.5 Kamera Time of Flight 23
5 Využité senzory a procesy pro bezkontaktní měření překážek a snímání polohy 25
5.1 Time of Flight senzor 25
5.1.1. Vzdálenostní měření za pomocí ToF senzoru 26
5.1.2. Měření rychlosti ToF senzorem 28
5.2 Arduino UNO R3 28
5.3 Ultrazvukový měřič vzdálenosti HC-SR04 29
5.4 RPlidar A1 30
5.5 Instalace ROSu 32
5.5.1. Vytvoření pracovního prostředí Catkin 33
5.6 RViz: 34
6 Zprovoznění a výsledné zaznamenávání hodnot pro určení polohy. 35
6.1 Návrh systému 35
6.2 Použitý robot 36
6.3 Pohon a řízení mobilního robota 37
6.4 Vytvoření podkladů pro uchycení komponentů. 39
6.5 Blokové schéma zapojení 40
6.6 Programovací část. 41
6.7 Ověření senzoru pro přesnost měření 42
6.8 Vyhodnocení měření ultrazvukovým senzorem 52
6.9 Servomotor a ultrazvukový senzor 53
6.10 Získána data a měření polohy robota 55
6.11 Lidarové vykreslení prostoru a určení polohy 56
7 Závěr 60
8 Literatura 61
Seznam příloh 64

Seznam zkratek a symbolů

CW	Continuous-wave
C	Kontrolní signál
c	Rychlost signálu
d	Vzdálenost
EKF	Extended Kalman Filter
F	Frekvence
f_r	Frekvence odraženého signálu
f_t	Frekvence vysílaného signálu
INS	Inerciální Navigační Systém
ICP	Iterative Closest Point
IR	Infrared
Q	Elektrický náboj
ROS	Robot Operating Systém
SLAM	Simultaneous Localization And Mapping
T	Perioda
ToF	Time of Flight
t	Čas
t_L	Zpoždění
USB	Universal seriál Bus
X	Vzdálenost
Δt_L	Nejmenší úsek času
Φ	Fázový posun

Seznam obrázků

Obrázek 1 – Trilaterace 2D [5] 11
Obrázek 2 – Nákres navigace výpočtem [7] 13
Obrázek 3 – Chybové elipsy během navigace [8] 14
Obrázek 4 – Schéma pro řešení Slamu 13
Obrázek 5 – Princip funkce optického senzoru. [9] 19
Obrázek 6 – Světelné závory (jednocestné) [9] 19
Obrázek 7 – Reflexní senzor (polarizační) [9] 20
Obrázek 8 – Potlačení pozadí [9] 20
Obrázek 9 – Difuzní reflexní senzor [9] 20
Obrázek 10 a 11 – Princip radaru [18] 22
Obrázek 12 – Detektor barvy TCS23 [14] 23
Obrázek 13 – Měření doby pulzního signálu mezi odeslaným a přijatým signálem [22] 24
Obrázek 14 – Měření fázového rozdílu mezi odesílanýma přijímaným signálem [22] 25
Obrázek 15 – Vyslání a příjem signálu [30] 26
Obrázek 16 – Spojitý signál TOF (odesílaný a přijímaný). [21] 27
Obrázek 17 – Pulzní signál TOF (odesílaný a přijímaný). [21] 27
Obrázek 18 – Vývojová deska Arduino UNO [19] 28
Obrázek 19 – Ultrazvuk pro měření vzdálenosti HC-SR04 [2] 30
Obrázek 20 – Znázornění velikosti detekčního úhlu senzoru plochy pro 15° [2] 30
Obrázek 21 – RPlidar A1 systémové komponenty [4] 31
Obrázek 22 – Příklad mapového prostředí při skenování RPlidar A1 [4] 31
Obrázek 23 – Blokové schéma systému 2D mapování, měření polohy a zaznamenávání trasy robota 36
Obrázek 24 – Použitý robot 36
Obrázek 25 – Grafové vyobrazení spolehlivosti v závislosti na síle signálu a vzdálenosti. [27] 38
Obrázek 26 – Rozměry hlavní podkladové desky 39
Obrázek 27 – Plechový úchyt pro boční senzory – rozměry 39
Obrázek 28 – Plechový kryt – rozměry 40
Obrázek 29 – Blokové schéma zapojení HC-SR04 s Arduinem 40
Obrázek 30 – Blokové schéma zapojení Arduino radaru 41
Obrázek 31 – Programovací kod pro jeden senzor HC-SR04 42
Obrázek 32 – Graf znázorňující chyby měření vzdálenosti proti sklu 44
Obrázek 33 – Graf znázorňující chyby měření vzdálenosti proti plechu 46

Obrázek 34 – Graf znázorňující chyby měření vzdálenosti proti plastu 48
Obrázek 35 – Graf znázorňující chyby měření vzdálenosti proti dřevu 50
Obrázek 36 – Graf znázorňující chyby měření vzdálenosti proti látce. 52
Obrázek 37 – Měření vzdálenosti za pomocí servomotoru a ultrazvuku 53
Obrázek 38 – Vytvořený kód pro Funci arduina 54
Obrázek 39 – Blokové schéma programu Processing 54
Obrázek 40 – Vykreslení naměřených pozic robota 55
Obrázek 41 – Lidarové vykreslení prostoru 56
Obrázek 42 – Lidarové 2D vykreslení prostoru 57
Obrázek 43 – Formát rozložení dat z lidaru [37] 58
Obrázek 44 – Lidarové vykreslení za pomocí Raspberry Pi 3 59

Seznam tabulek

Tabulka 1 – Radarová pásma a jejich využití [17] 22
Tabulka 2 – Specifikace snímače HC-SR04 [2] 30
Tabulka 3 – Technické parametry motoru [26] 37
Tabulka 4 – Technické parametry regulátoru [25] 37
Tabulka 5 – Technické parametry přijímače [28] 38
Tabulka 6 – Naměřené hodnoty od senzoru ke skleněné překážce 43
Tabulka 7 – Naměřené hodnoty od senzoru ke plechové překážce 45
Tabulka 8 – Naměřené hodnoty od senzoru ke plastové překážce 47
Tabulka 9 – Naměřené hodnoty od senzoru ke dřevěné překážce 49
Tabulka 10 – Naměřené hodnoty od senzoru ke látkové překážce 51
Tabulka 11 – Hodnoty získané ze senzorů v centimetrech 55
Tabulka 12 – Hodnoty získané z lidaru v mm 56

1 Úvod

Tato bakalářská práce se zabývá popisem měřením polohy mobilního robota v budově a nástroji umožňující měření polohy. Měření polohy je jeden z hlavních problému v mobilní robotice a v dnešní době se jí věnuje pozornost.

Práce je rozčleněna do několika kapitol s tím, že první kapitola je úvodní a poslední obsahuje závěr. Teoretická část je zaměřena v první řadě na samotnou lokalizaci, kde je popsána jak lokalizace absolutní, tak relativní. Následně je teoretická část zaměřena na tvorbu mapy a na snímače umožňující měření polohy robota.

Praktická část se nejprve zaobírá výběrem a popisem vhodných senzorů pro realizaci měření. Pro tuto práci byly vybrány Ultrazvukové měřiče vzdálenosti a Lidar. Následně je zde obsažen návrh systému pro měření polohy mobilního robota, vykreslení prostoru a vykreslení cesty. To vše za pomocí Lidaru, ultrazvukových senzoru, Arduino desek, Raspberry Pi 3 a počítače s potřebnými programy. Následně tato část obsahuje samotnou realizaci.

2 Lokalizace

Během návrhu pohybového mobilního robota vznikají základní problémy, mezi které patří například, jakým způsobem má být zjišťována aktuální poloha robota a určení samotná cílová poloha, kterou má daný robot dosáhnout. V neposlední řadě řeší způsob, kterým se na požadované místo může dostat. Zjišťováním aktuální polohy se zabývá právě lokalizace. Bez znalosti aktuální polohy mobilního robota nemůžeme plánovat jak samotnou trasu, tak ani cíl dané cesty.

Určování polohy můžeme rozdělit například podle různých kritérií na metody globální lokalizace a lokální lokalizace. Lokální lokalizace zabezpečuje sledování okolního prostoru robota. Dodává informace o směru a vzdálenosti překážky, která se vyskytuje v okolí. Globální lokalizace nám slouží k určení pozice daného robota v prostoru. Dané metody se dále dělí na absolutní a relativní lokalizace. Tyto dvě metody popisuji v další části textu práce. Každá ze zde zmíněných metod má své pro a proti. Z tohoto důvodu se ve valné většině aplikací využívá jejich kombinace.

2.1 Absolutní lokalizace

Absolutní lokalizace slouží k jednoznačnému určení absolutní polohy robota v prostoru, jako je například místnost nebo budova. Daná lokalizace nezávisí na čase, protože zde nedochází ke sčítání chyb. To je zapříčiněno tím, že lokalizace není závislá na předchozí poloze. U tohoto druhu lokalizace je chyba měření v konstantním rozmezí pro veškeré naměřené polohy. Nejčastějšími využívanými metodami jsou triangulace a trilaterace.

2.1.1. Triangulace

U této metody určení neznámé polohy mobilního robota se využívá nejméně tří úhlů, mezi referenčními body a robotem. Na mobilním robotu je umístěn rotační snímač, který získává informace o úhlu mezi viditelným snímačem a podélnou osou robota. Pokud jsou tyto dané body známy, tak s jejich pomocí v globálním souřadném systému bude mobilní robot schopen vypočítat svou danou polohu pomocí změřených úhlů. Referenčním bodem je často používán vysílač infračerveného záření, dále například akustické vlny nebo elektromagnetické záření.

Ideální by bylo využití všesměrového vysílače. Daný vysílač je bohužel energeticky náročný, především v momentě, kdy je potřeba robota lokalizovat ve velkém prostoru. Kvůli tomu se upřednostňují vysílače s kuželovitou vyzařovací charakteristikou.

Existují také další možné realizace. Jedná se například o rotační vysílač/přijímač, který je umístěn na robotovi a v jeho blízkosti se umísťují reflexní plochy u kterých je známá poloha. Reflexních ploch musí být v robotově blízkosti tři a více.

2.1.2. Trilaterace

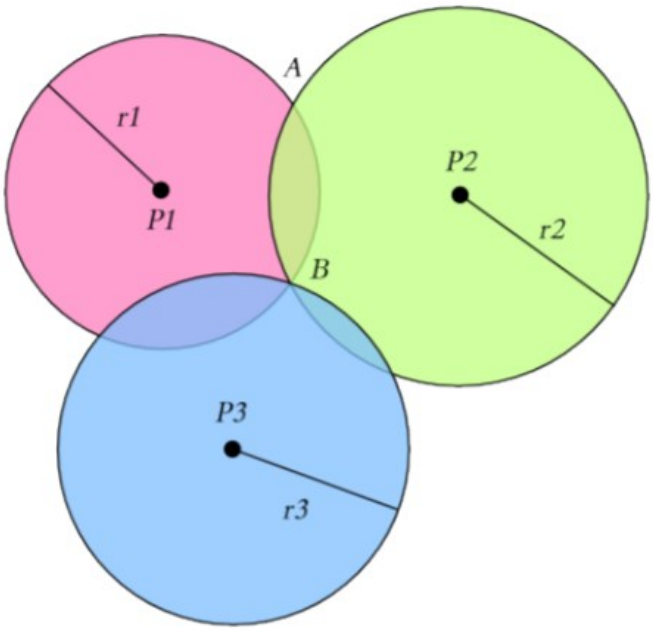
Jedná se o metodu pro určování polohy na základě měření nebo pomocí výpočtů vzdálenosti mezi referenčními body a robotem. U této metody musí být použity také minimálně tři referenční body jako u metody triangulace. Tyto body jsou umístěny na neznámých místech v pracovním prostoru.

Nalézá se zde několik realizačních možností. Jedná se například o použití referenčních bodů jako vysílače a umístění přijímače na robota nebo v opačné verzi, kdy jsou umístěny v pracovním prostoru přijímače se známou polohou a jeden vysílač na robotovi. Při umístění přijímače na robotovi vzniká výhoda, díky které je možno určit polohu více robotů v jeden časový okamžik.

2.1.3. 2D trilaterace

Jedná se o lokalizaci v dvourozměrném prostoru, kde musí být robot společně s referenčními body umístěn v jedné rovině. Určují se zde souřadnice x a y, které udávají informaci o přesné poloze mobilního robota v pracovním prostoru. K výpočtu polohy danou metodou se musí určit tři vzhlednosti mezi robotem a referenčním bodem.

Na obrázku 1 lze vidět princip 2D Trilaterace pomocí grafického znázornění. V grafickém znázornění se nacházejí referenční body, které jsou označeny P1, P2, P3. Kolem jednotlivých bodů je kružnice, která je určena poloměry r1, r2, r3. Velikost r1, r2, r3 je určena naměřenou vzdáleností od robota. Poloha mobilního robota je v obrázku označena bodem B, kde se nachází průsečík kružnic.



Obrázek 1: Trilaterace 2D [5]

2.1.4. 3D trilaterace

3D trilarerace se využívá pro zjištění polohy v trojrozměrném prostoru. Oproti 2D trilareraci je zde rozdíl v tom, že k určování polohy robota můžou být referenční body umístěny v rozdílné výšce. Toto je způsobeno tím, že při této metodě se nevyužívají kruhy jako u metody 2D, ale jsou zde použity koule, u kterých se také vyhledávají průsečíky.

2.2 Relativní lokalizace

Při této lokalizační metodě je lokalizace robota určována z relativní změny polohy mobilního robota vůči jeho dané předcházející poloze. Je zde zaměřovaná rotační a polohová změna, dále případně rychlost a zrychlení, ze kterých je možnost dopočítat změnu polohy. Vypočítána výsledná poloha mobilního robota se určí pomocí složení dílčích změn. Pracuje se s dílčími změnami od počátku samotného měření.

K přesnému určení relativní polohy při měření je zapotřebí změřit naprosto přesně změnu polohy v každém časovém okamžiku. Tento úkol je pro nás v samotné praxi relativně těžký a z tohoto důvodu je odhadovaná výsledná poloha robota zatížená chybou. Tato zatěžovací chyba se sčítá při každém jednotlivém měření polohy. Z tohoto lze usoudit, že metodu relativní lokalizace u mobilního robota není vhodno využívat během dlouhého časového úseku, během kterého je sledovaná poloha robota.

Prostředky relativní lokalizace nejčastěji používají senzory, které je potřeba kalibrovat. Pro změnu kalibrace je možné v některých případech využít porovnání výsledku z jednotlivých lokalizačních metod (absolutní a relativní). Společným principem veškerých prostředků je určení změny pozice. To je prováděno integrováním jednotných dílčích změn pozice robota. Během způsobu realizace těchto postupu se veškeré metody odlišují. Jednotlivé změny pozice mohou být:

- Naměřeny přímo při odometrii
- Vypočítány pomocí integrací okamžité rychlosti v čase

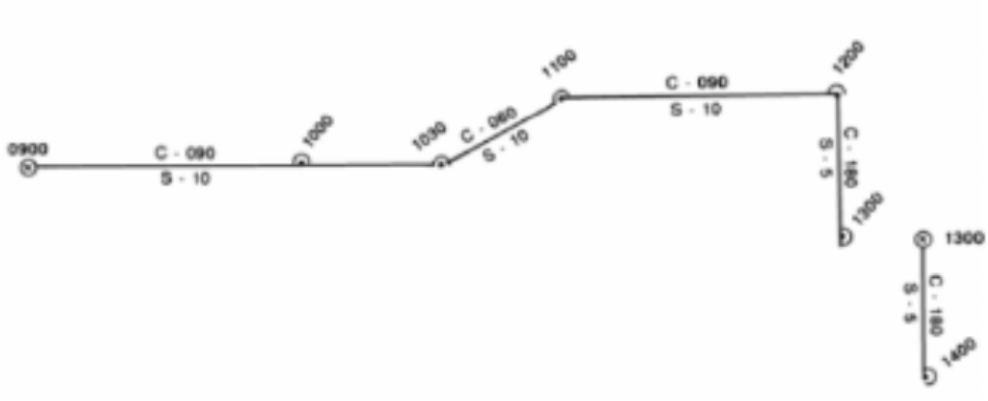
Několikanásobné integrování způsobuje, že i malá nepřesnost nebo šum je v průběhu měření příčinou chyby výsledné hodnoty, která se stále zvětšuje s časem.

Co je tím prostředkem relativní lokalizace? Zde se jedná o tzv. Dead Reckoning. Tento název je odvozen z deducated reckoning (lokalizace výpočtem).

2.2.1. Dead Reckoning

Jedná se o velmi jednoduchou výpočetní metodu, ze které odhadneme aktuální pozici robota z jeho předchozí polohy. Pro daný výpočet se využívají data o směru pohybu, rychlosti a času, který uplynul od poslední dané pozice. Tato starší metoda se již dříve využívala ve středověku u námořních plaveb nebo například později také v začátcích letectví společně s přesnějšími metodami. Mezi tyto metody patří navigace podle hvězd nebo orientačních bodů, které jsou viditelné (tzv. landmarků). [23]

Termín „Lokalizace výpočtem“ je občasně využíván jako společný název pro veškeré metody relativní navigace. Nejvyužívanější metodou, která je do této skupiny zařazena je odometrie. Odometrie požaduje k určení polohy pouze matematický model robota a informaci ohledně natočení kol.



Obrázek 2: Nákres navigace výpočtem. Pozice se zakresluje v pravidelných časových intervalech. Při každé změně rychlosti nebo kurzu, opravě polohy nezávislou lokalizační technikou. Převzato z [7]

2.2.2. Odometrie

Odometrie je složená ze dvou slov. Hodos (cesta) a metron (měřit), která jsou původem z řečtiny. Odometrie je metoda, která nám popisuje datovou transformaci. Data jsou poskytována rotačními enkodéry, které slouží k měření otáčení kol běžných nebo hnacích. Rotační enkodér převádí rotační pohyb na elektrický signál, který je spravovatelný. Díky své spolehlivosti se v mobilní robotice využívají enkodéry reflexní a transmisivní, které patří mezi optické. Transmisivní enkodér využívá přerušování světelného paprsku, který prochází děrovaným diskem. Reflexní enkodér využívá naopak plný disk s matnými a reflexními plochami.

Veškeré údaje, které z enkodéru dostáváme obsahují chybu, která je zanedbatelná. Tyto chyby nemají vliv na výsledek, pokud je měření v malém počtu. V momentě, kdy se měření pohybují ve větším počtu akumulují se chyby, které při každém jednotlivém měření vznikají. Chyby dělíme na systematické a náhodné. Systematické chyby mají při opakování měření naprosto stejnou velikost. Pokud ne, jejich velikost se mění předvídatelně a minimálně.

Systematické chyby rozdělujeme na:

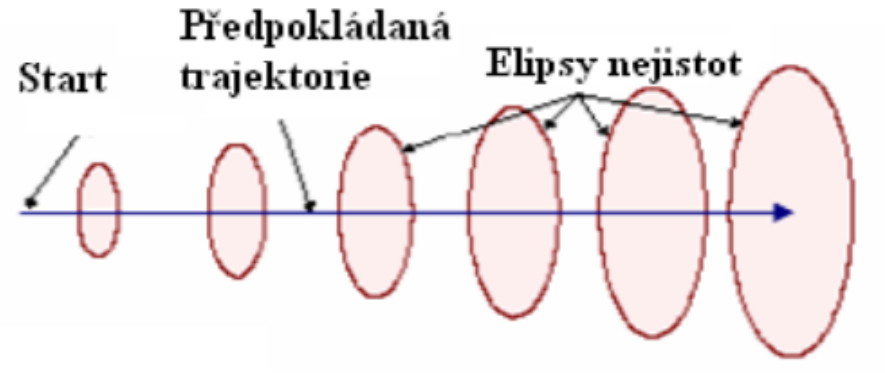
- Rozdílná velikost kol.
- Nekruhový tvar kol.
- Rozdíl mezi skutečným a nominálním rozměrem kol.
- Konečné rozlišení enkodéru.

Nahodilé chyby se v průběhu měření nepředvídatelně mění a nemůžeme je odstranit korekcí.

Náhodné chyby rozdělujeme na:

- Pohon na nerovném povrchu.
- Pohon přes nepředvídatelné předměty v trase robota.
- Prokluz kol způsobenými okolními vlivy:
    - Kluzký povrch.
    - Nárazem.
    - Natočení kol.

Tyto chyby způsobují časté využívání Dead Reckoningu. Vypočtená pozice robota je obklopena chybovou elipsou, která se s delší dráhou, kterou robot urazí zvětšuje. Viz obr. 3. [24]



Obrázek 3: Chybové elipsy během navigace [8]

2.2.3. Inerciální určování polohy

Využívá akcelerometry a gyroskopy k měření podélného a také rotačního zrychlení, jejichž integrováním získáme rychlost a následně i ujetou vzdálenost.

Inerciální navigace je soběstačná a platí u ní stejný nárust chyb jako u odometrie v závislosti na čase prováděného měření. Při této metodě navigace vznikají další chyby, se kterými jsme se u odometrie nesetkali, protože tyto chyby jsou zapříčiněny gyroskopem a akcelerometrem.

Dané senzory mají určitou citlivost a rozlišení. To způsobuje problém týkající se měření zrychlení v průběhu měření, pokud se robot pohybuje pomalu. Chyba tohoto měření nastává tehdy, pokud je velikost zrychlení robota podobna velikosti šumu, který je způsobován nedokonalostí senzorů.

Mezi další problém můžeme zařadit gyroskop. Problém nastává tehdy, pokud gyroskop udržuje svou orientaci pouze vůči sobě samému a neudržuje svou orientaci vůči zemi. Je způsobováno rušení, které způsobuje rotace země kolem slunce, ale také i rotace kolem své osy.

Můžeme se dále setkat s gravitačním zrychlením, které nám také způsobuje potíže. Toto je zapříčiněno akcelerometrem. U svislé změny polohy mobilního robota akcelerometr podává více

informací, nikoli pouze informace o zrychlení robota samotného. Výstupem samotného akcelerometru je součet navigačního zrychlení společně se zrychlením robota.

Velkou výhodou inerciálního určování polohy je jeho nezávislost na prostoru, kde se robot pohybuje.

2.2.4. INS s pevnou orientací akcelerometrů v prostoru

Tento systém využívá mechanické gyroskopy. Tyto gyroskopy jsou tvoří Kardanový závěs a setrvačník.

Kardanový závěs umožnuje setrvačníku zachování jeho orientace, a to i bez ohledu na rotaci mobilního robota. Pokud chceme, aby mechanická gyroskop fungoval co nejdéle, musíme udržovat aktivní otáčky setrvačníku. Na aktivních otáčkách setrvačníku závisí přesnost mechanického gyroskopu. Tato přesnost je také závislá na závěsu a minimálním tření ložisek v setrvačníku. Pokud chceme přenášet signál z akcelerometru například přes pohyblivé závěsy musíme použít sběrací kroužky.

Pro vyhodnocení změny polohy robota z naměřených rychlostí využíváme pevnou orientaci akcelerometrů. Kvalitní gyroskopy jsou kvůli svému složitému mechanickému provedení velice drahé.

K selhání mechanického gyroskopu může dojít tehdy, pokud při manévrech dojde k natočení prstenců závěsů, které poté otočí silně osou setrvačníku.

2.2.5. INS s akcelerometry pevně spojitými se zařízením

Tato možnost je levnější kvůli jejímu jednoduchému mechanickému provedení. Při této levnější variantě musíme ovšem také měřit úhlové rychlosti otáčení společně se zrychlením. Je zde nutno přepočítávat změřená zrychlení podle právě aktuálního natočení akcelerometrů do souřadného systému. Přesnost záleží na využité technologii. Kruhové laserové gyroskopy mají velmi vysokou přesnost a jsou mnohem lepší variantou než vysoce kvalitní mechanické gyroskopy.

Využívají se zde gyroskopy pro měření relativního otáčení robota nikoli k udržení stálé orientace. Tyto gyroskopy nám dodávají informace o rychlosti otáčení kolem osy (jedné). Pracují například na těchto fyzikálních principech:

- Optické gyroskopy, které využívají Sagnacova jevu. Sagnacův jev je závislost interference dvou paprsků, které procházejí kruhovou cestou oběma směry na fyzické rotaci.
- Piezoelektrické měří Coriolisovu sílu. Když se gyroskopem otáčí, daná síla působí na vibrující součást gyroskopu. Dané senzory jsou vyráběný v MEMS provedení (Mikro Elektro-Mechanické Systémy).

2.3 ROS

ROS (The Robot Operating System) jedná se o framework, který slouží k vytváření aplikací pro roboty. ROS je sbírka knihoven a nástrojů, které mají za cíl zjednodušit úkol vytváření komplexních a robustních chování robotů přes širokou škálu robotických platforem.

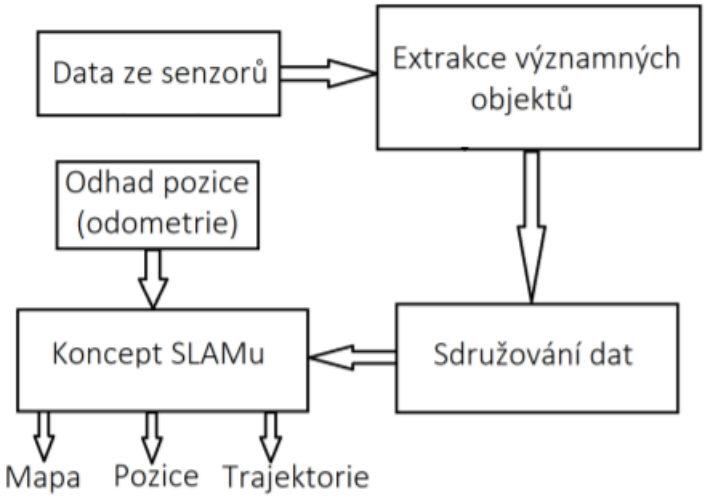
Díky flexibilitě mohou uživatelé využít jen požadované části ROSu a zbytek si dodělat sami. Finální program je složen z jednotlivých uzlů neboli procesů. Komunikace uzlů mezi sebou probíhá za pomocí zpráv. Rozdělení na uzly nám umožnuje využití jednoho kódu pro několik aplikací či spuštění uzlů na rozličných počítačích. Systém dělíme mezi dvě vrstvy, tj. robotický software, který vytvořila komunita ROS a jádro.

3 SLAM

SLAM neboli Simultaneous Localization and Maping je soustředěn na tvorbu map z neznámého prostředí a aktuální lokalizaci v mapě. [31] Mapa je tvořena pomocí různých senzorů, které pozorují okolí a odhadují svůj vlastní stav. Tento proces je základní pro roboty, kteří se pohybují v neznámém prostředí, kde nemáme data o současné poloze robota. [32] Tato oblast robotiky je usilovně zkoumána.

SLAM implementujeme několika způsoby, díky využití různých softwarových a hardwarových prostředků. Není to algoritmus ale koncept, který používá rozlišné způsoby pro dosažení cíle, kterými jsou tvorba mapy a správná lokalizace. [31]

Tento proces nám lépe přiblíží následující obrázek.



Obrázek 4: Schéma pro řešení Slamu.

Nejdříve tento algoritmus získává data z robotova okolí. Nejčastěji používáme za tímto účelem dva senzory: kameru a laserový skener. Kamera je náročnější pro zpracování, ale přinese nám více informací než laserový skener. Problém nastává při zhoršených světelných podmínkách nebo při úplné tmě. Pokud bychom nechtěli nebo nemohli využít kameru využili bychom druhý typ senzorů, kterým je 2D Lidar. [31] Výhoda tohoto Lidaru je nenáročné zpracování a přesnost, naopak nevýhoda spočívá v omezeném využití při větší koncentraci částic ve vzduchu (kouř, prach). Během robotova pohybu se data, která odesílá senzor změní. Díky této změně můžeme vypočítat informace o pohybu. Tyto data nám výsledný odhad zpřesní nebo alespoň přispějí k získání tohoto odhadu.

3.1 Landmark extraction

SLAM algoritmus, který je založen na Lidaru je složen z několika částí. Jednou z těchto částí je extrakce významnějších objektů z měřených dat. Zde se jedná o podstatná místa, podle kterých se později robot dokáže orientovat. Jedná se o různé geometrické útvary, rohy v dané místnosti a mnoho dalších částí, které nezapadají mezi zbytek získaných dat. Pro vyhledání těchto bodů můžeme využívat různé metody. V následující kapitole si popíšeme základní dva typy.

3.1.1. Spikes algoritmus

Tento algoritmus vyhledává extrémy a hroty. Vyhledává ve spojitém celku hodnoty, které jsou odlišné. Musí být stanoveny podmínky pro význačný bod a pro rozlišení šumu. Problém v tomto algoritmu způsobují lidé a objekty, které se pohybují. Tyto objekty algoritmus musí rozpoznat a nesmí je zařadit mezi význačné rysy. Pokud dojde k tomu, že algoritmus objekt označí jako významný, způsobí to v pozdější práci znehodnocení výsledku. Pro možnost přiřazení objektů do významných rysů měl by daný objekt být odlišitelný daty od ostatních, statický a také opakovaným měřením změřitelný. [31]

3.1.2. Sliding window algoritmus

Sliding window algoritmus, plovoucí okno, pracuje se třemi body a slouží pro vyhledání rohů. Tento algoritmus si vybírá lichý počet bodů, ze kterých následně vybírá prostřední a krajní body. Velikost okna se ve většině případů volí v rozmezí 11 až 15 vzorku. Mezi prostředním a krajním bodem se sestrojují dva vektory a mezi nimi se vypočte úhel. [33] Úhel, který je menší než 120° je považován za roh.

3.2 Data association

V následující části se budeme věnovat asociaci dat, která spočívá ve vyhledávání už objevených význačných bodů v novém pozorování. [31]

Nejdříve nemáme žádná data, a proto musíme vytvořit databázi význačných bodů. Do databáze zařazujeme jen takové body, které vidíme N^1 abychom minimalizovaly chyby. Všechny nově příchozí význačné body sdružujeme s ostatními body, které již v databázi máme. Body přiřazujeme pomocí algoritmu. Nejjednoduššími algoritmy jsou ICP [34] nebo nearst-neighbor approach [31]. Jestliže nový bod je stejný jako význačný bod z databáze navýší se počet pozorování tohoto bodu, pakliže význačnému bodu neodpovídá, označíme jej jako nový význačný bod a navýšíme počet pozorování.

3.3 SLAM koncept

3.3.1. EKF SLAM

Extended Kalman Filter neboli EKF. Zde je prováděno odhadování pozice libovolného systému nebo daného robota za pomocí rekurze. Odhad o novém stavu robota, jeho polohy obdrží na základě dat odometrie, kterou dále upřednostňujeme za pomocí Kalmanova filtru během nového pozorování. S počtem měření vzrůstá výpočetní náročnost [35].

3.3.2. FastSLAM

Je založen na nezávislém měření význačných bodů a znalosti pozice robota. Známá poloha prvního význačného bodu nám neposkytuje informaci o následujících význačných bodech [36]. SLAM problém je transformován pomocí FastSLAMu na výpočet polohy robota z dílčích význačných bodů. [35] Přesná pozice je následně udána pomocí nejvyšší pravděpodobnosti polohy robota, kterou určíme pomocí nezávislého měření význačných bodů, tzn. FastSLAM pracuje s množinou pravděpodobných trajektoriích [36].

4 Snímače pro lokalizaci

4.1 Optický senzor

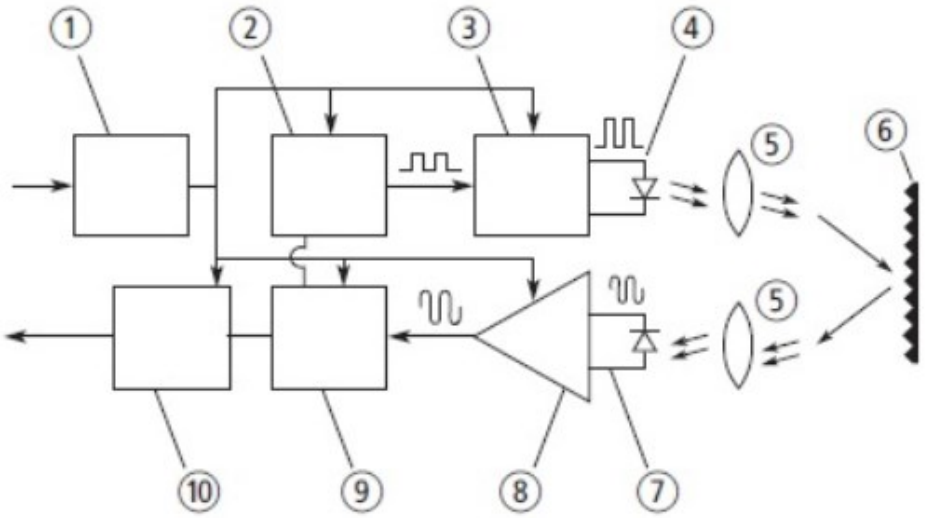
K měření vzdálenosti a detekci překážek se využívají senzory, které přeměňují energii elektrickou na světlo. Odrážené světlo je zpětně převáděno na elektrickou energii. Světlem je myšleno elektromagnetické vlnění, které má vlnovou délku od ultrafialového záření až k infračervenému záření. Jako zdroje záření se využívají laserové diody, infračervené diody a také i LED diody. Mezi přijímače tohoto záření patří například:

- Fotodiody.
- Fototranzistory.

Ty mají za úkol převést přijatý optický signál na energii elektrickou. Díky možnosti mít přijímač a vysílač v jednom zapouzdření máme možnost detekovat překážky, které se během cesty nacházejí a určit vzdálenost překážky od senzorů. Princip IR senzorů se zakládá na detekování intenzity odrazovaného světla. Příklady využití optických senzorů:

- Možnost počítání jednotlivých objektů na pohyblivém pásu.
- Možnost zastavení provozu v momentě, když senzor zjistí hromadění produktů.
- Při balení: senzory kontrolují naplnění, zabalení a označení nálepkami na balících.
- Možnost detekovat chyby pásu.
- Možnost zjistit, zda je potřebný materiál při výrobě k dispozici.

Princip funkce:

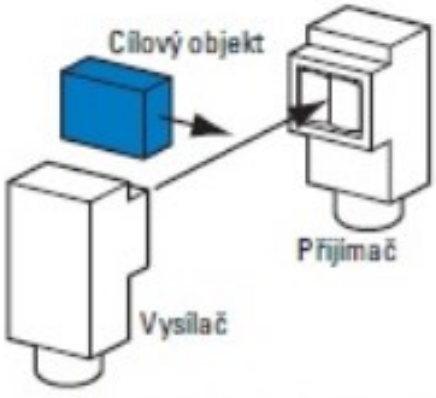


Obrázek 5: Princip funkce optického senzoru. [9]

1) Zdroj napájení, 2) Modulátor, 3) zesilovač, 4) Světelná (LED) dioda, 5) Čočka, 6) Odrazové nebo cílové pole, 7) Přijímač, 8) Zesilovač příjmu, 9) Demodulátor, 10) Výstupů.

Pracovní režimy:

1) Světelné závory (jednocestné):

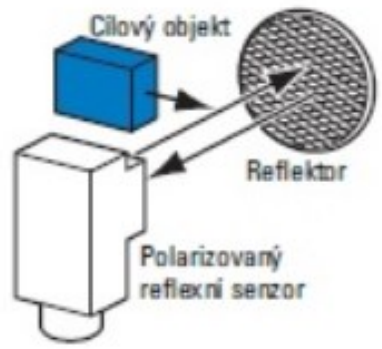


Popis:

Světelný paprsek je vysílán od vysílače k přijímači. Pokud dojde k jeho přerušení znamená to, že se cílový objekt pohybuje mezi vysílačem a přijímačem.

Obrázek 6: Světelné závory (jednocestné) [9]

2) Reflexní senzor (polarizační):

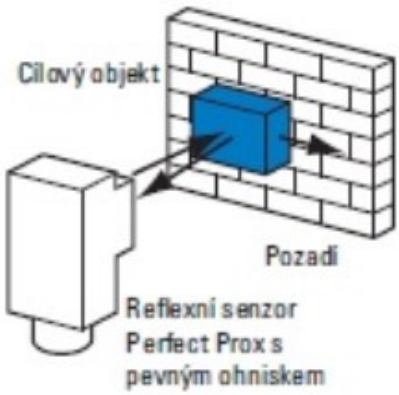


Popis:

Zde můžeme vidět jedno zapouzdření kde je umístěn jak přijímač, tak vysílač. Světelný paprsek je otáčen o 90°a je odrážen zpět za pomocí polarizační odrazové plochy, která slouží jako odrazová.

Obrázek 7: Reflexní senzor (polarizační) [9]

3) Potlačení pozadí:

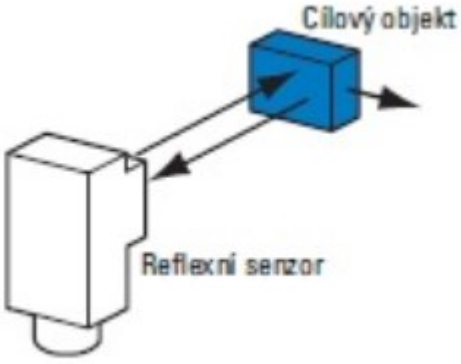


Popis:

Zde se jedná o speciální difúzní senzor, který je složen ze dvou přijímačů. Tento senzor snímá projekty v definované vzdálenosti a umí také potlačovat objekty, které se nacházejí mimo danou oblast.

Obrázek 8: Potlačení pozadí [9]

4) Difuzní reflexní senzor:



Popis:

Stejně jako u polarizačního reflexního senzoru je i zde jedno zapouzdření ve kterém se nachází vysílač i přijímač. Pokud se před senzorem pohybuje objekt je paprsek odrážen zpětně k danému přijímači.

Obrázek 9: Difuzní reflexní senzor [9]

4.1.1. IR senzor

IR nebo také jinak infračervený senzor. Tyto senzory jsou nejčastěji využívány v robotice k detekci libovolných překážek v blízkosti mobilního robota. Je možno je také využít pro vzdálenosti od překážek. Záření s vlnovou délku delší než světlo, které je viditelné je emitováno IR LED diodou. Pro možnou detekci se využívají fotodiody nebo fototranzistory. Jedná se o fotodiody, které jsou citlivé na světlo s danou vlnovou délkou. Na výstupu z přijímačů dostáváme informaci, která nám sděluje, jestli světlo, které je emitované IR LED diodou se odráží zpětně k přijímači. Zjišťujeme tak, zda se nachází nějaké překážka v okolí mobilního robota.

K zjištění přesné polohy překážky, která se nachází v prostoru je zapotřebí více vysílačů. Díky nim poznáme, kde se daná překážka nachází a to tak, že od vysílačů se odráží světlo a po zpětném dodání informací zjistíme v jakém směru se překážka nachází. Síla odraženého světla je velmi ovlivňována zbarvením povrchu dané překážky. Tuto vlastnost je možno využít na mobilním robotovi pomocí senzoru detektor barvy. Tento senzor umožní robotovi jet po různě zbarvené čáře na podlaze. Pokud čáru odlišíme na několika místech barevně můžeme robota naprogramovat tak, že na různé barvě změní některou ze svých vlastností.

Samotný IR senzor je možno využít také například pro měření vzdálenosti mezi robotem a překážkou. K tomuto měření se využívá například citlivost přijímače nebo princip triangulace. Jako u jiných snímačů se i zde mohou vyskytnou chyby způsobené měřením. Chyba může vzniknout například tak, že se v blízkosti mobilního robota nachází překážka/y, které pohltí záření. Jedná se o překážky s tmavým povrchem, na které záření dopadá. Dále můžeme také narazit překážku/y, které mohou mít velice lesklý povrch a tento povrch způsobí to, že odrazí záření mimo přijímač.

4.1.2. Laserový senzor

Tyto typy senzoru jsou velice drahá varianta pro měření. Jejich výhodou je ovšem velice přesné měření. Tento typ senzorů funguje na stejném principu jako senzor infračervený. Nalezneme v nich ovšem jeden rozdílný prvek. Jedná se o druh světelného signálu, který je v tomto případě laserová dioda. U laserové diody máme možnost využít měření na delší vzdálenosti mezi robotem a překážkou. Možnost měření delších vzdáleností je zapříčiněno vyšší intenzitou světla. Laserové senzory jsou ideální například pro navádění vozidel, automatizací, řízení dopravy.

4.2 Ultrazvukový senzor vzdálenosti

Tento typ senzorů se v průmyslu vyznačuje svou velkou spolehlivostí a také univerzalitou. Vzhledem k tomu, že se tato metoda měření funguje za všech podmínek můžeme ji například využít k měření hladiny na přesnost milimetrů nebo například k detekci objektů. Měření tohoto senzoru funguje tak, že vyhodnotí čas mezi vysílaným signálem a jeho přijetím. Tento čas se nejen u tohoto senzorů označuje zkratkou ToF. Ta pochází z anglického spojení Time of Flight. Výslednou hodnotou tohoto měření je vzdálenost od nejbližšího umístěného objektu, od kterého se vyslaný signál odrazí zpět. U tohoto měření jsme schopni dosáhnout velice přesných výsledků, protože se akustický signál šíří vzduchem pomalu. Ultrazvuk je tlumen ve vzduchu a jeho tlumení stoupá v momentě kdy roste frekvence daného ultrazvuku mimo kapaliny. Mezi nevýhody tohoto měření patří například krátká

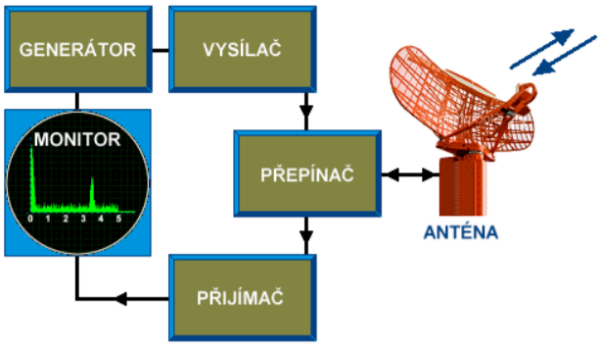
vzdálenost měření nebo velká prodleva mezi danými měřeními. Zdrojem chyb měření u ultrazvukového senzoru vzdálenosti může být:

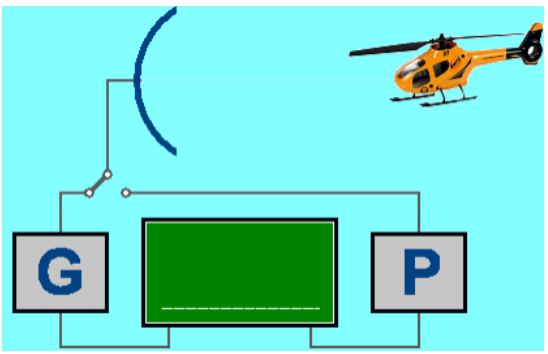
- Rozdílná rychlost šíření vln během různých teplot vzduchu
- Několika násobný odraz vlnění, signálů od okolních materiálů pohlcujících zvuk.

4.3 Radar neboli radiolokátor

Radar neboli radiolokátor pochází z anglického Radio Detecting And Ranging. Jedná se o zařízení, které slouží k zaměření objektů a určení jejich vzdálenosti. Radar je vysílač a zároveň přijímač rádiových vln, a to nejvíce v pásmu mikrovln.

Princip radaru: Radar je založen na odrazu rádiových vln. Generátor vytváří vysokofrekvenční signál, ten přes přepínač putuje do antény. Po odvysílání je přepínač přehozen do stavu příjmu. Přijímač po určitý časový úsek snímá úroveň signálu z antény. Výstup je poté zobrazován na monitoru, intenzita signálu odpovídá intenzitě svitu monitoru (analogový radar). [17]





Obrázek 10 a 11: Princip radaru [18]

Vzdálenost obou výchylek na vodorovné stopě je přímo úměrná vzdálenosti, z jaké se vrací odražený impulz. Tak je možno na stupnici určit okamžitou vzdálenost letounu od antény. Radary pracují v určitých pásmech s danou frekvencí a vlnovou délkou. [18]

Tabulka 1: Radarová pásma a jejich využití [17]

Pásmo	**Frekvence (GHz)**	**Vlnová délka (cm)**	**Využití**
L	1-2	15-30	Řízení leteckého provozu
S	2-4	7,5-15	Řízení leteckého a námořního provozu
C	4-8	3,75-7,5	Satelitní vysílání
X	8-12	2,5-3,75	Sledování a řízení raket, námořní radary, meteorologie, letecký provoz, radarové mapování
K	12-40	0,75-2,5	Radarové mapování, satelitní navigace, meteorologie, měření rychlosti, letecký provoz
mm	40-300	0,1-0,75	Trojrozměrné mapování

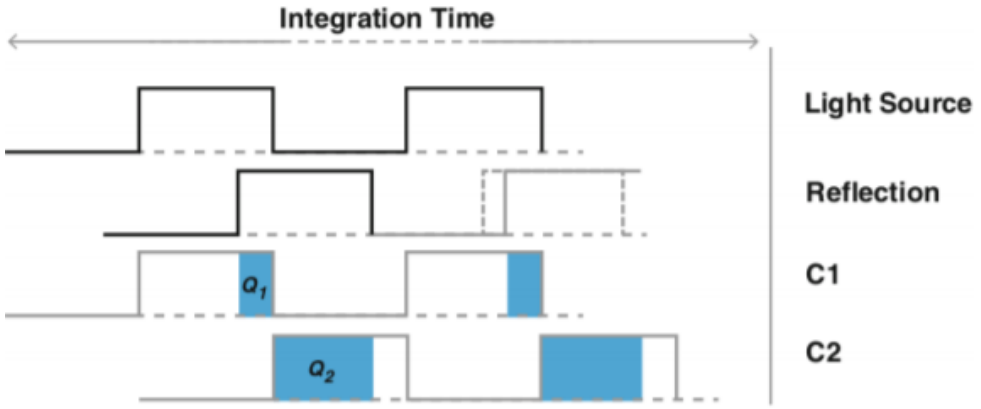
4.4 Detektor barvy

Umožnuje detekovat barvu v jeho blízkosti. Je složen ze senzorů, který se skládá z bílých diod sloužící pro osvětlení a z fotodiod s barevnými filtry. Složení těchto součástech na senzoru nám umožnuje zjistit barvu předmětu. Po osvětlení za pomocí bílých diod se světlo odrazí zpětně k senzoru. Po nakonfigurování jednotlivého senzoru je senzor schopen generovat frekvenci výskytu jednotlivých barevných složek (RGB).[14]

Obrázek 12: Detektor barvy TCS23. [14]

4.5 Kamera Time of Flight

Time of Flight kamera, označovaná zkratkou ToF kamera, využívá měření principem „doby letu“. TOF kamera je složena ze snímače, který snímá elektromagnetický signál a vysílače. Vysílač zde zastupuje infračervená LED dioda, která osvěcuje prostor signálem. Infračervené záření, které je odráženo je následně měřeno maticovým senzorem. Pro každý pixel v tomto senzoru je měřena doba t mezi odesílaným a přijímaným signálem. Následně se dopočítává vzdálenost d, která se vypočítá pomocí vzorce 4.1. Vzhledem k tomu, že vzdálenost 1 mm odpovídá 6,6 ps (pikosekunda) je požadováno velice rychlé elektroniky.



Obrázek 13: Měření doby pulzního signálu mezi odeslaným a přijatým signálem. [22]

Infračervená LED dioda jakožto světelný zdroj vysílá spojitou vlnu (CW) continuous-wave, která je modulovaná čtvercovým signálem nebo jednotlivé pulzy. Na obrázku 12 můžeme vidět jakým způsobem je u pulzního zdroje odměřován čas. Infračervená LED dioda svítí dobu Δt. Kamerovým řídícím obvodem jsou vytvořeny kontrolní signály označeny jako C1 a C2 s opačnou fází trvající dobu Δt. V průběhu kontrolních signálu jsou hromaděny elektrické náboje Q_1 a Q_2 na fotodiodách daného snímače. Ty jsou využity pro výpočet podle rovnice:

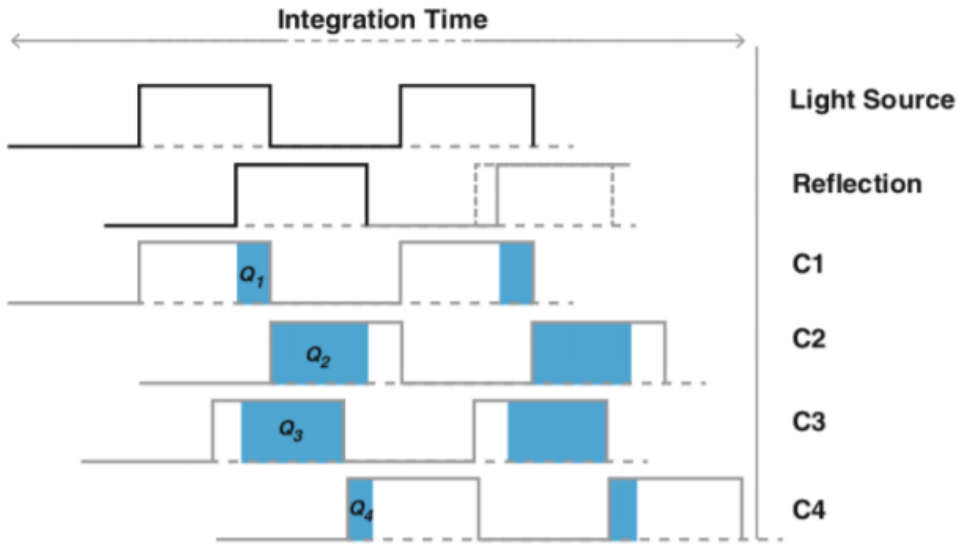
$$d = \frac{\Delta tc}{2}\frac{Q2}{Q1+Q2} \tag{4.1}$$

Time of flight kamery se signály CW měří vzdálenost za pomocí měření fázového posunu. Využívají se čtyři kontrolní signály. Ty jsou posunuty fázově o 90°, viz obrázek 13. Hodnotami z nábojů Q_1 až Q_4 lze dopočítat vzdálenost a fázový posun za pomocí vztahů 4.2 a 4.3. [22]

$$d = \frac{c}{4\pi f}\Phi \tag{4.2}$$

$$\Phi = arctan\frac{Q3-Q4}{Q1-Q2} \tag{4.3}$$

V odraženém signálu se odstraní za pomocí rozdílu $Q_3 - Q_4$ a $Q_1 - Q_2$ konstantní offset. Konstantní offset je způsoben například okolním světlem, které se vmísí do již odraženého signálu. Během výpočtu dále dojde k standardizaci amplitud a to tak, že výpočet fáze není závislý na offsetu a tlumení amplitud je daleko větší než u pulzní kamery. Kamera CW je ovšem omezována maximálním fázovým posunem, který je 2π. Tomu je odpovídající max. vzdálenost $d\max = c/2f$. Proto se více využívají světelné zdroje multispektrální. [22]



Obrázek 14: Měření fázového rozdílu mezi odesílaným a přijímaným signálem. [22]

5 Využité senzory a procesy pro bezkontaktní měření překážek a snímání polohy

Tato kapitola nám nabízí přehled nejčastěji využívaných bezkontaktních senzorů pro detekci překážek. Tyto senzory dělíme na dva typy: optické a neoptické. Neoptické senzory k detekci překážek užívají elektromagnetického záření, jehož vlnová délka je mimo viditelné a také mimo infra červené spektrum. Jako příklad optických senzorů můžeme uvést technologie radaru, jenž užívají rádiové vlny či sonar, která užívá zvukové vlny. Optické senzory užívají elektro magnetické záření o vlnových délkách blízkého infračerveného tj. (0,7 – 1000µm) a viditelného tj. (400 - 700nm) spektra. Následně je dělíme na aktivní a pasivní. Pasivní senzory užívají odraz okolního světla od měřeného objektu. Jako příklad nám může sloužit obyčejná kamera. Aktivní senzory na rozdíl od pasivních mají zdroj záření, jenž po odrazu od měřeného objektu zachytávají a získávají tak informaci o vzdálenosti tohoto objektu a jeho struktuře.

Pro možnost měření polohy mobilního robota v budově byly použity komponenty Arduino UNO R3, sloužící pro naprogramování a získávání hodnot. K tomuto bude sloužit program Arduino. Dále bude využito šest ultrazvukových měřičů vzdálenosti HC-SR04, které jsou připojeny na desku Arduino. Tyto snímače nám měří vzdálenost mezi senzorem a překážkou nebo stěnou v dané místnosti. Výsledné hodnoty budeme zobrazovat ve využitém programu od Arduina. Pro měření a zmapování prostředí nám bude sloužit RPlidar A1.

5.1 Time of Flight senzor

Jedná se o senzor, který nám udává měřenou hodnotu vzdálenosti mezi veškerými předměty, které jsou v úhlu záběru daného senzoru a sebou samým. Toto měření je prováděno ultrazvukem nebo světelnými paprsky. Častěji využívanou metodou k měření je zde světelné záření. Je to způsobeno tím, že světelné záření nám poskytuje široké pásmo přenosu a vyšší rychlost, než je tomu u ultrazvuků. Tehdy, když je využívána infračervená část spektra dochází ke snižování šumu. Když je snížen šum je

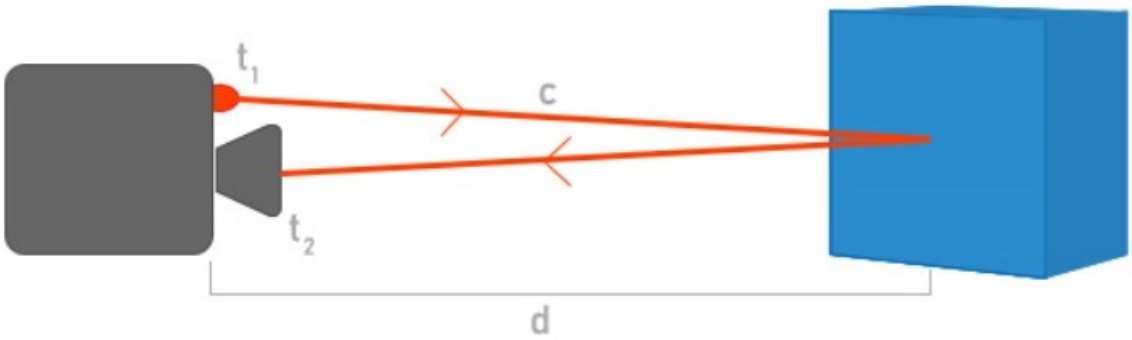
možno lépe odlišit paprsek od denního světla. K těmto senzorům se řadí především sonar, lidar, radar a také TOF kamera.

Využití Time of Flight senzorů:

- Skenování předmětů.
- Navigace robotů uvnitř budov.
- Detekce překážek.
- Úpravu digitálních fotografii v režimu real-time.

Funkce Time of Flight senzorů:

Z daného senzoru je vyslán signál, který se následně od daného předmětu (překážky) odrazí zpětně k přijímači, který je umístěn v senzoru. Čidlo zachytí odražený signál, který se nachází v úhlu záběru čidla. Pro stanovení vzdálenosti předmětu je zapotřebí porovnat dvě věci, a to čas kdy byl daný signál vyslán a čas přijetí odraženého signálu.



Obrázek 15: Vyslání a příjem signálu. [30]

5.1.1. Vzdálenostní měření za pomocí ToF senzoru

ToF senzory vysílají signály, které dělíme na spojité a pulzní. Při vysílaných spojitých signálech ze senzoru měření vzdálenosti můžeme provádět fázovým posunem, který změříme mezi signálem odesílaným a přijímaným. Z následujícího obrázku 16 můžeme pozorovat, že přijímaný signál sinusového průběhu obsahuje zpoždění t_L Vzhledem k tomu, že známe periodu T , je hodnota t_L přímo úměrná Φ (fázovému posunu). Danou hodnotu tedy vypočítáme za pomocí vzorců:

$$t_L = \frac{\Phi}{2\pi} \mathrm{T} = \frac{\Phi}{2\pi} \frac{1}{f} \tag{5.1}$$

f = frekvence vysílaného signálu. Změřená vzdálenost x a rozlišení této vzdálenosti Δx je následně podle 5.3:

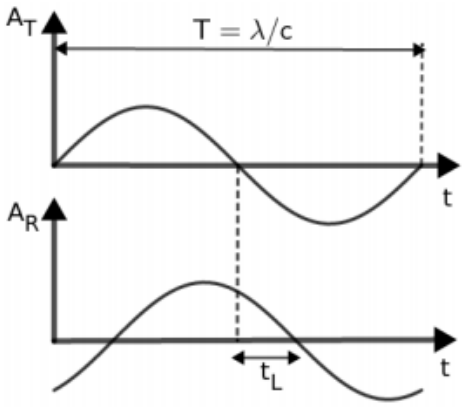
$$\mathrm{x} = \frac{c}{4\pi \mathrm{f}} \Phi \tag{5.2}$$

$$\Delta x = \frac{c}{4\pi f}\Delta\Phi \tag{5.3}$$

Rozlišení senzorů, které využívají spojité signály, je závislé nejen na schopnosti využité elektroniky, ale i na frekvenci signálu, který je využíván. Velkou nevýhodou těchto signálu je, že fázový posun je maximálně 2π. Proto je největší možná měřená vzdálenost:

$$x_{max} = \frac{c}{4\pi f}2\pi = \frac{c}{2f} = \frac{\lambda}{2} \tag{5.4}$$

Toto je důvod, proč se u těchto senzorů využívají signály složené z většího počtu sinusoid, které mají různé vlnové délky. Během tohoto postupu platí, že nejmenší vlnová délka udává rozlišení a maximální vzdálenost nám sděluje největší vlnová délka. [21]



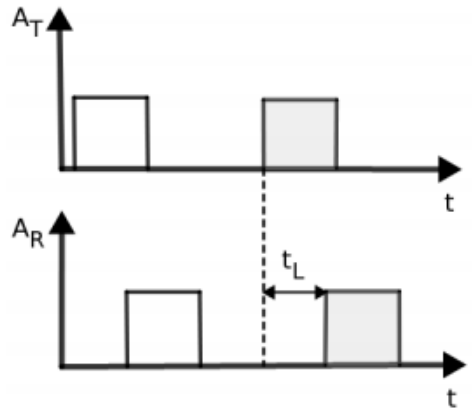
Obrázek 16: Spojitý signál ToF (odesílaný a přijímaný). [21]

Pulzní senzory měří čas, který je mezi poslaným a přijatým signálem. Na obrázku 17 vidíme zobrazení časového průběhu amplitud odesílaného (A_T) a přijímaného (A_R) signálu. Jedná se o signály pulzní. Změřený čas t_L lze převést na vzdálenost následujícím převodním vztahem:

$$d = \frac{t_L c}{2} \tag{5.5}$$

kde c = rychlost signálu a d = vzdálenost ve které se signál odráží. Z této rovnice také můžeme odvodit Δd rozlišení, kde je Δt_L nejmenší úsek času, který je tento systém umožnuje rozlišit [21]:

$$\Delta d = \Delta t_L \frac{c}{2} \tag{5.6}$$



Obrázek 17: Pulzní signál TOF (odesílaný a přijímaný). [21]

5.1.2. Měření rychlosti ToF senzorem

ToF senzory měříme vzdálenost, ale některými těmito senzory můžeme měřit také rychlost pohyblivých objektů. K tomuto je využíván Dopplerův jev. Pro možnost měření rychlosti musí daný senzor umožnovat měření frekvence odraženého signálu. Při pohybu objektu nebo senzoru bude frekvence signálu (f_r), který se odráží, rozdílná oproti frekvenci vysílaného signálu (f_t). Vztah mezi těmito frekvencemi s očekávaným pohybem objektu, který má rychlost pohybu v směrem k vysílači daného signálu je tento vztah vyjádřen rovnicí 5.7:

$$f_r = f_t \frac{\mathrm{c+v}}{\mathrm{c-v}} \tag{5.7}$$

Kde c = rychlost šíření signálu. Rozdíl mezi těmito dvěma frekvencemi f_ra f_t se jmenuje Dopplerův posun. Ted si můžeme s použitím vzorce 5.7 vyjádřit jako

$$f_d = f_r - f_t = f_t \frac{2\mathrm{v}}{\mathrm{c-v}} \tag{5.8}$$

Tato rovnice se dá ještě zjednodušit. Musí být ovšem splněna podmínka, že v << c.

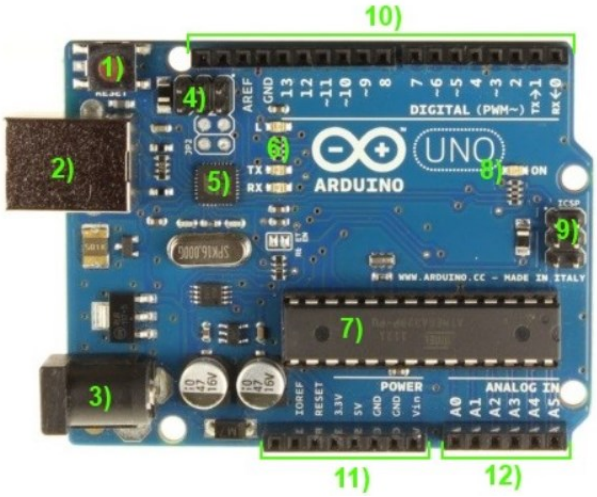
$$f_d = f_t \frac{2\mathrm{v}}{\mathrm{c}} \tag{5.9}$$

Z předešlé rovnice 5.9 je snadné vyjádřit rychlost:

$$v = \frac{\mathrm{c} f_d}{2 f_t} \tag{5.10}$$

Během pohybu daného objektu v opačném směru by došlo k prohození znamének v rovnici 5.7 ve jmenovateli a čitateli. Ovšem na výsledku rovnice 5.10 by nenastala žádná změna. [21]

5.2 Arduino UNO R3



Obrázek 18: Vývojová deska Arduino UNO [19]

Arduino UNO je vývojová deska, kterou v tomto případě využijeme pro propojení, naprogramování a čtení hodnot ze senzorů. Tato deska se skládá hned z několika komponentů, které si popíšeme.

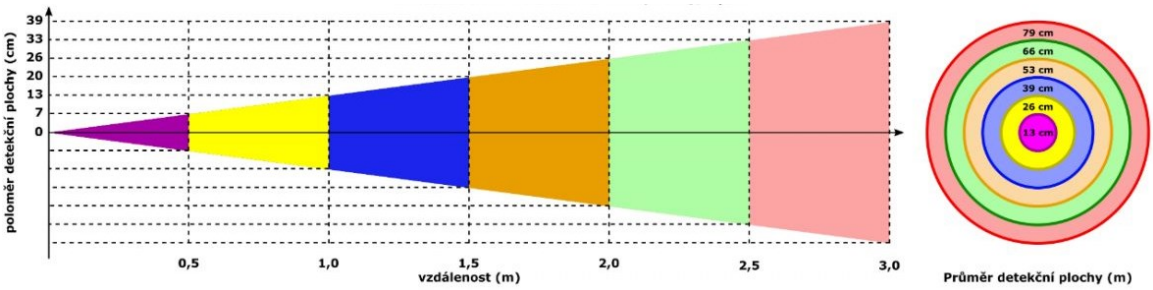
1. Resetovací tlačítko – slouží pro možnost spuštění programu od samotného začátku.
2. USB konektor – jedná se o konektor typu B, který slouží pro propojení desky s počítačem.
3. Napájecí konektor – je využívaný pouze tehdy, pokud Arduino není napájeno přes USB.
4. ICSP hlavice – umožnuje externě naprogramovat USB-serial převodník.
5. USB-serial, převodník – tento převodník slouží pro komunikaci hlavního čipu s počítačem.
6. LED diody – nalezneme zde typy L, TX, RX. Diody, které jsou označeny jako RX a TX jsou aktivní, pokud je komunikace vedena pomocí sériové linky.
7. Mikrokontroler ATmega 328 – zde je nahrán kód bootloader (zavaděč). Do tohoto nahráváme program z počítače a ten pomocí svých vstupů a výstupů dále vše ovládá.
8. LED dioda – jedná se o indikační diodu s označením ON, která je aktivní, pokud je připojeno napájení.
9. ICSP hlavice – umožnuje externě naprogramovat hlavní čip.
10. Digitální piny – pro vstupy a výstupy
11. Napájecí výstupy a vstupy
12. Analogové vstupy – slouží pro možnost připojení vodičů, díky kterým chceme měřit analogové hodnoty.

5.3 Ultrazvukový měřič vzdálenosti HC-SR04

Jedná se o ultrazvukový měřič vzdálenosti, který je často využíván u robotů a na místech je potřebujeme prozkoumat prostor před daným senzorem. Tento snímač má velice velkou přesnost detekce, která se nachází v rozmezí od dvou centimetrů do čtyř metrů. Tato přesnost se ovšem zhoršuje od 2 metrů. Jeho velkou výhodou je nízká spotřeba a vysoká přesnost měření.

Funkcí tohoto modelu je sepnutí modulu TRIG pomocí desky Arduino po dobu 5 mikrosekund. Vysokofrekvenční pulz je vyslán ultrazvukovým vysílačem a po vyslání čekáme na zachycení pulzů pomocí ultrazvukového přijímače. Po zachycení přečteme délku impulzů, kterou obdržíme z výstupu ECHO. Délku impulzů převedeme konstantou na vzdálenost udávanou v centimetrech. Konstanta je vypočtena vydělením rychlosti vzduchu při 20° desetitisíci > obdržíme cm/mikrosekundu. Z důvodu cesty tam a zpět výsledek vydělíme dvěma. Když obrátíme hodnotu výsledku po zaokrouhlení obdržíme číslo 58,31, kterým dělíme obdržený čas. Pro měření vzdálenosti při jiné teplotě nebo atmosféře musíme dohledat rychlost vzduchu v dané atmosféře pomocí fyzikálních tabulek.[3]

Obrázek 19: Ultrazvuk pro měření vzdálenosti HC-SR04 [2]



Obrázek 20: Znázornění velikosti detekčního úhlu senzoru plochy pro 15°. [2]

Tabulka 2: Specifikace snímače HC-SR04. [2]

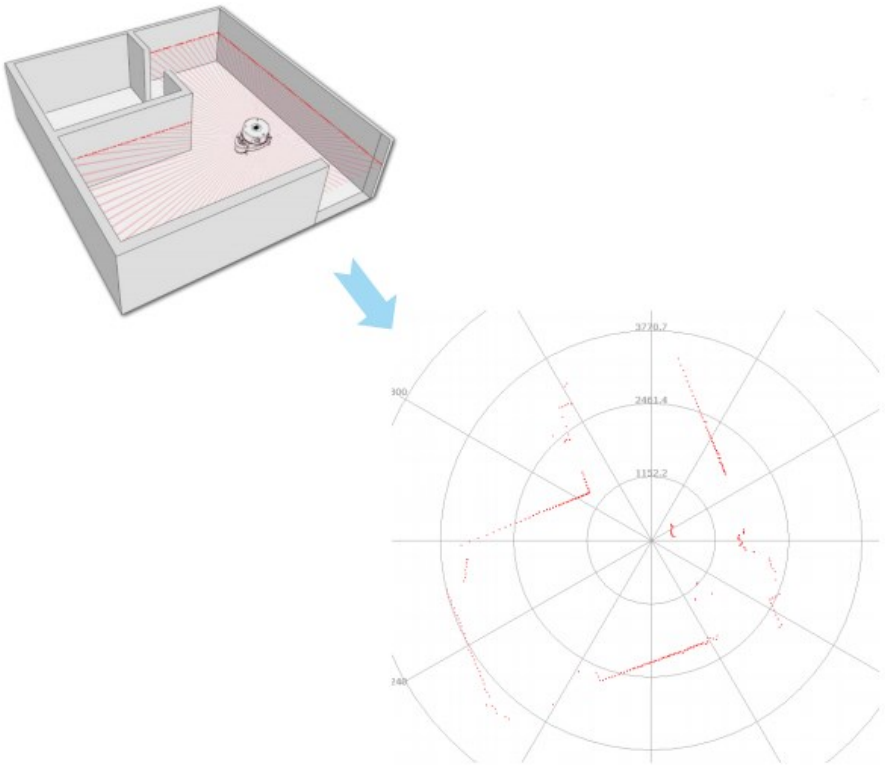
Čipy	TL074, STC11, MAX323	**Zorný úhel**	15°
Provozní napětí	5VDC	**Rozsah měření**	2-400 cm
Proud	2 mA	**Rozlišení**	3 mm
Frekvence ultrazvukového signálu	40kHz	**Rozměry (mm)**	40 x 20 x 1,6

5.4 RPlidar A1

Jedná se o 360stupňový 2D laserový skener. Dokáže provádět skenování v rozsahu 6 metrů v 360-ti °. Vytvořená 2D data v úložišti mohou sloužit k mapování, lokalizaci nebo také k modelovaní prostředí. V průběhu při vzorkování po 360-ti bodech dosáhla frekvence skenování 5,5 Hz a maximální hodnota konfigurace činí 10 Hz. RPlidar A1 se při své funkci otáčí po směru hodinových ručiček a v tomto směru také skenuje. Rozsah dat můžeme získat za pomocí USB. Systém lidaru upraví frekvenci

automaticky podle dané rychlosti motoru. Je založený na principu laserové triangulace a využívá vysokou rychlost hardwaru. Údaje o vzdálenosti měří 2000krát za sekundu.

Obrázek 21: RPlidar A1 systémové komponenty. [4]



Obrázek 22: Příklad mapového prostředí při skenování RPlidar A1. [4]

RPlidar A1 můžeme například využít k:

- Lokalizaci a navádění čistícího robota.
- Lokalizace chytrých hraček a vyhýbaní se překážkám.
- Modelování a skenování okolního prostředí.

5.5 Instalace ROSu

Co je to ROS jsme si shrnuli v krátkosti v části 2.3 a nyní se zaměřme na instalaci. Instalace bude provedena za pomoci konzolových příkazů v terminálu, které se provádějí v Ubuntu. Můžeme také rovnou vzít složku s obsahem a jednotlivě instalovat jednotlivé části což je však zdlouhavý a komplikovanější postup. Nejprve je nutno povolit v nastavení Ubuntu možnost stahovat software přes internet z důvodů bezproblémové instalace. Přesněji je nutno povolit universe, multiverse a restricted.

Zbytek se děje již pouze v terminálu. Nejdříve počítači povolíme příjem softwaru z webové stránky packages.ros.org. ROS pracuje výhradně s Ubuntu 16.04 a Ubuntu 15.10. Příkaz, který spojuje balíčky je následující.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc
/apt/sources.list.d/ros-latest.list'
```

Následně je ihned provedena konfigurace kláves.

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1C
F6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Musíme se ujistit, že máme aktualizované balíčky případně je musíme aktualizovat.

```
sudo apt-get update
```

Protože ROS má mnoho nástrojů a knihoven máme ve výběru několik možností, které lze stáhnout podle žádaného obsahu. Je doporučováno Desktop-Full, které obsahuje v podstatě vše, co je potřeba tj. ROS, rviz, 2D a 3D simulátor, navigaci a 2D/3D vykreslování, rgt.

```
sudo apt-get install ros-kinetic-desktop-full
```

Verze Desktop je celkem obsáhlá i když neúplná. Je složena z ROS, RViz, robotické knihovny a rqt.

```
sudo apt-get install ros-kinetic-desktop
```

Další variantou je ROS-Base, což je základní verze obsahující jen communication libraries, package, build.

```
sudo apt-get install ros-kinetic-ros-base
```

Poslední variantu máme stažení konkrétního balíčku, které provedeme pomocí příkazu.

```
sudo apt-get install ros-kinetic-PACKAGE
```

PACKAGE nahradíme jménem balíčku, plánujícího se stáhnout. K vyhledávání balíčku, které jsou dostupné nám poslouží příkaz:

```
apt-cache search ros-kinetic
```

Dříve než použijeme ROS musíme inicializovat rosdep. Díky rosdepu můžeme instalovat systémové závislosti pro zdroj, které si rosdep žádá zkompilovat. Ovšem rosdep je také velmi důležitým ke správnému chodu některých částí jádra ROS.

```
sudo rosdep init
rosdep update
```

V následujícím kroku instalace nastavíme pracovní prostředí, což je výhodné pro situaci, kdy je proměnné prostředí ROS přidáno automaticky do bash relace vždy při zpuštění nového shellu.

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Když potřebujeme změnu prostředí pouze pro aktuální shell můžeme napsat:

```
source /opt/ros/kinetic/setup.bash
```

Tímto poslední krokem končí instalace. V následujícím kroku se nastaví pracovní prostředí catkin workspase.

5.5.1. Vytvoření pracovního prostředí Catkin

K tvorbě vlastní aplikace ve frameworku ROS, je třeba nejdříve vytvořit uživatelské pracovní prostředí. Catkin je součástí Rosu od verze Groovy a dále, který se zpřístupní pomocí příkazu

```
sudo apt-get install ros-kinetic-desktop
```

Následně se podíváme na nastavení prostředí. Nejdříve si uděláme složku catkin_make a otevřeme jí.

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

Pracovní prostoj můžeme stále vytvářet, přestože je tento pracovní prostor zcela prázdný. (v adresáři 'src nejsou žádné balíčky, pouze jediný odkaz CMakeLists.txt).

```
$ cd ~/catkin_ws/
$ catkin_make
```

Catkin_make je ideálním nástrojem pro práci s pracovním prostorem catkin. Aktuální adresář by měl obsahovat složky 'devel' a 'build'. V devel složce se následně vytvoří několik souborů * sh. Před pokračováním je nutno nechat rozběhnout zcela nový soubor setup.*sh.

```
$ source devel/setup.bash
```

Pokus správně překryjeme pracovní prostor díky instalačnímu scriptu musíme zajistit, že proměnná prostředí ROS_PACKAGE_PATH má adresář v němž se nacházíte. Pokud vše funguje instalace byla dokončena.

5.6 RViz:

Během vývoje našeho robota či jeho ovládání je dobré mít vyobrazen grafický výstup, k čemuž nám slouží RViz. Jedná se o vizuální nástroj zobrazující data posílaná snímači robota a informace o stavu, které posílá ROS.

Pokud chceme spustit RViz musíme nejprve zpustit ROS. Následně Rviz otevřeme příkazem:

```
rosrun rviz rviz
```

6 Zprovoznění a výsledné zaznamenávání hodnot pro určení polohy.

6.1 Návrh systému

Návrh systému pro určení polohy mobilního robota je složen z několika částí. První část je věnována ultrazvukovým senzorům, které měří polohu robota od stěn nebo překážek. Tyto senzory budou naprogramovány a budou jednotlivě vypisovat hodnoty vzdálenosti v reálném čase. Další část systému bude obsahovat měření polohy robota a vzdálenosti od překážek v uhlu od 0° do 180° servomotorem a ultrazvukem za pomocí prostředí od Arduina a pomocí Processingu bude umožněno vykreslování v reálném čase. Další podstatnou částí navrženého systému bude RPLidar A1, který nám umožní za pomocí aplikace framegrabber Module od společnost Slamtec vykreslování místnosti, ve které se pohybuje, zobrazí polohu robotu a za pomocí ukazatele budeme schopni změřit vzdálenost od stěn. Dále bude tento senzor využit pro 2D mapování, měření polohy a zaznamenávání trasy robota za pomocí framework ROS a jeho částí Hector Slam. Při využití Raspberry Pi 3 bude vytvořena dálková komunikace například do počítače nebo tabletu za pomocí VNC viewer, které nám umožní vzdáleně vyobrazovat vykreslování za pomocí Lidaru a měření polohy robota využitím ultrazvukových senzoru. Použité komponenty a instalace ROSu jsou popsány v kapitole 5.

Popis mnou požívaného systému Hector Slam, který je součástí ROSu.

Hector Slam:

Nejdůležitějším rozdílem tohoto systému je schopnost mapování prostoru, bez použití odometrie nebo jinými senzory poskytujícími informace, které nám udávají polohu stroje. Při této metodě tvoří mapu pouze data poskytována laserovým skenerem. Z těchto důvodů musím použít zařízení, které umožnuje vysokorychlostně snímat okolí (hlavně lidar). Lidar nám dokáže snímat prostor několikrát za sekundu s rozlišením několika stovek bodů.

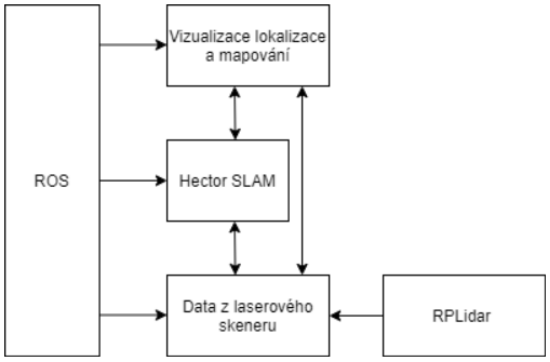
Princip a použití balíčku Hector:

V našem případě využíváme balíčku Hector Navigation, Hector Localization a Hector SLAM. Balíček Hector Localization používá k fúzi dat ze senzorů rozšířeného Kalmanova filtru, který jsem popsal v kapitole 3.3.1. Během tvorby mapy pomocí Hector SLAM se vypočítává poloha robota podle vyvážené mapy, kde jsou porovnávaný sledy skenů. Jestliže je mapa tvořena podle dat z Lidaru musíme robot ovládat podle principu kompozice mapy. Robot by se měl řídit způsobem, který umožní vyhnutí se ostrých úhlů, jenž by mohly mít vliv na nespojitosti ve vytvořené mapě.

Algoritmus balíčku odhaduje polohu v horizontálním prostoru díky porovnávaní koncového bodu paprsku a získané mapy. Koncové body se promítají už v obdržené mapě, kde se odhaduje pravděpodobnost obsazeného místa. Shodu skenu a aktuální mapy řeší Gauss-Newtonova rovnice, zajištující co nejlepší transformovanost skenovaných paprsků do mapy.

Klíčové zdrojové síťových uzlů (node) pro HECTOR SLAM jsou hector_trajecotry_server, hector_mapping a hector_geotiff. Pro obsluhu Lidaru se využívá urg_node. Node hector_trajectory_server slouží pro ukládání pohybové trajektorie robota. Vždy je nutno brát v úvahu transformace každého z rámců platformy, trajektorie se ukládá jako nav_msgs/Path. Dále z nodu máme hector_geotiff starající se o ukládání mapy a robotovy trajektorie do geotiff (soubor obrázků),

které udávají mřížku obsazenosti v rámci komponované mapy. Pro proceduru SLAM je hlavním nodem Hector_mapping.

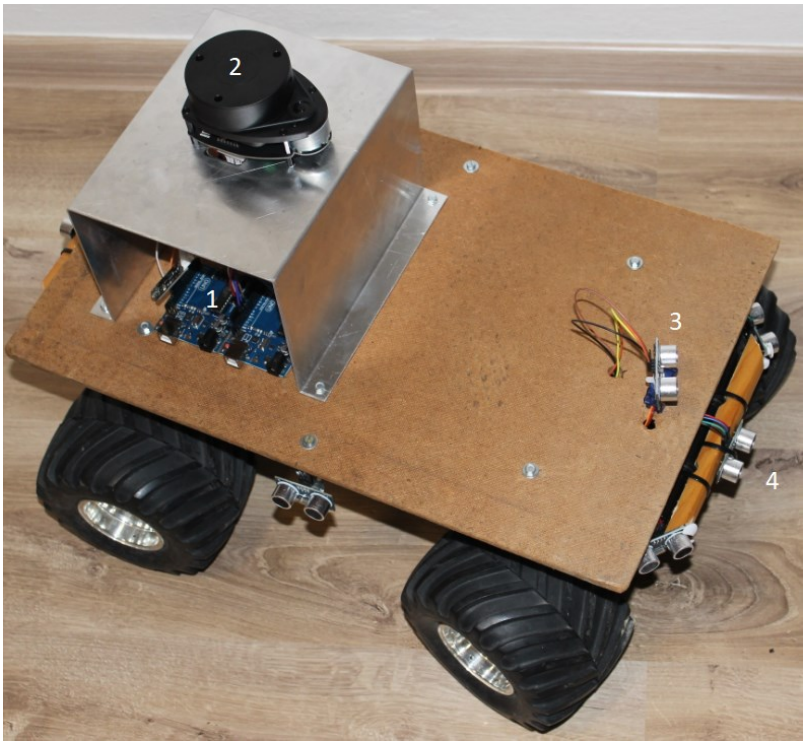


Obrázek 23: Blokové schéma systému 2D mapování, měření polohy a zaznamenávání trasy robota.

6.2 Použitý robot

V této kapitole si v krátkosti popíšeme využitého robota, který byl využíván v této práci pro měření polohy. Fotografii tohoto robota naleznete na obrázku 24 s popisem senzorů a důležitých komponentů. Rozměry tohoto robota včetně kol činí na délku 480 mm a na šířku 385 mm. Výška daného robota je 230 mm bez krytu. S krytem a lidarem robot měří na výšku 410 mm.

V této práci je využíván podvozek z mobilního robota Monster Pick-up TXT-1 s pohonem 4x4. Tento hliníkový podvozek s daným pohonem a nezávislým zavěšením jednotlivých kol nám umožnuje robotu překonávat také i obtížný terén.



Obrázek 24: Použitý robot: 1 – řídící část, 2 – Lidar, 3 – servomotor a ultrazvukový senzor, 4 – ultrazvukové dálkoměry.

6.3 Pohon a řízení mobilního robota

Pro možnost pohonu daného robota je využita 12 V baterie. Tato baterie pohání dva motory MIG 500 Turbo 7.2 V Race, elektrický regulátor rychlosti a následně také pohon natočení přední a zadní nápravy. Využité motory jsou navrženy jako vysokootáčkové. Motory mají zesílený magnetický plášť. Oba tyto motory jsou odrušeny za pomocí dvou kondenzátoru umístěných uvnitř těchto motoru a také jsou vybaveny ventilátorem, který zajištuje snižování teploty rotoru a zvyšuje tak účinnost těchto motorů.

Tabulka 3: Technické parametry motoru [26]

Napětí [V]	3,6 až 8,4
Jmenovité napětí [V]	7,2
Otáčky [ot/min]	22.000
Hmotnost (s / bez pláště) [g]	164 / 191
Průměr [mm]	36
Délka [mm]	50
Průměr hřídele [mm]	3,17
Volná délka hřídele [mm]	18,4

Využitý elektrický regulátor je vodotěsný i prachotěsný a můžeme jej tak využívat za každého počasí. Umožňuje obousměrnou regulaci a je navržen pro více stejnosměrných motorů v mém případě se jedná o dva motory. Regulátor je vybaven napěťovou ochranou, která během nízkého napětí odpojí motor dále také ochranou proti možnému přehřátí a samotným odpojením během ztráty signálu.

Tabulka 4: Technické parametry regulátoru. [25]

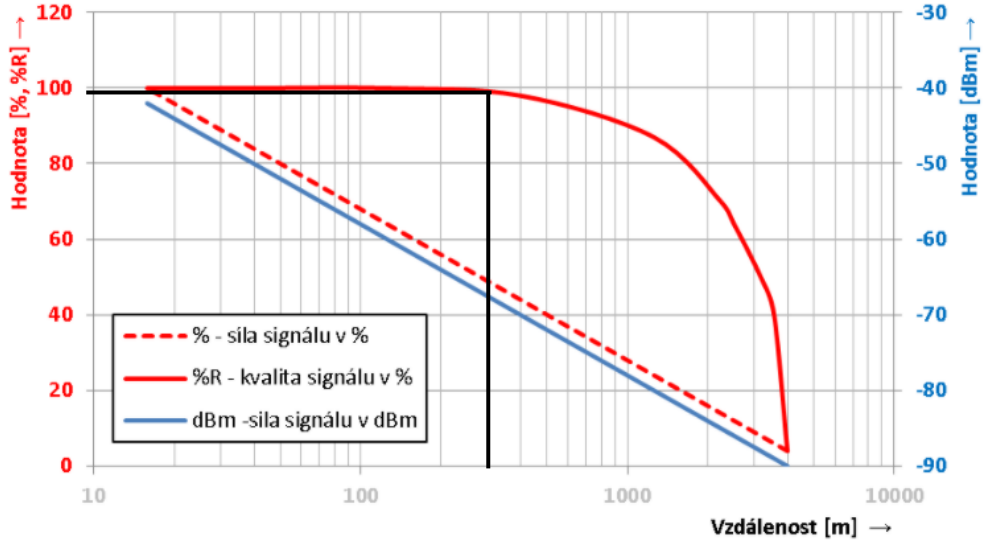
Použití	Auto
Typ	stejnosměrný
Trvalý proud [A]	80
Max. špičkový proud [A]	400
Délka [mm]	46
Šířka [mm]	35
Výška [mm]	26.5
Hmotnost [g]	75
Regulace	obousměrná
Brzda	Ano
Vstup pro senzory	Ne
Programovatelný	Ano

Na kostře robota je umístěn a připojen čtyř kanálový přijímač (SPEKTRUM AR400 DSM2). Tento přijímač slouží pro komunikaci a umožnění vzdáleného ovládání mobilního robota za pomocí ovladače SPEKTRUM DX6i.

Tabulka 5: Technické parametry přijímače [28]

Pásmo	2.4 GHz
Modulace	Spektrum DSM2, DSMX
Napájecí napětí	3.5 až 9.6 V
Počet kanálu	4

V následujícím grafu je vyobrazena spolehlivost v závislosti na síle signálu a vzdálenosti. V obrázku 25 je vyznačena závislost pro vzdálenost 300 metrů, která je pro mé využití naprosto dostačující.

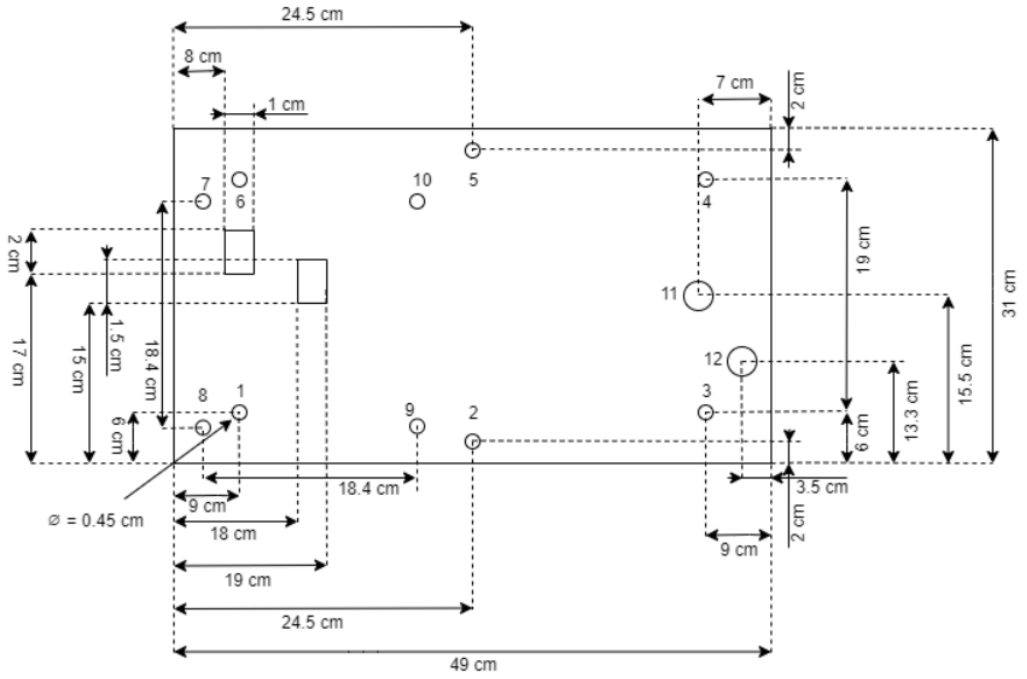


Obrázek 25: Grafové vyobrazení spolehlivosti v závislosti na síle signálu a vzdálenosti. [27]

Poslední potřebnou částí pro možnost ovládání je ovládací prvek SPEKTRUM DX6i. Využívá se v pásmu 2,4 GHz. Tento modem má velice přesnou synchronizaci pohybu serv a pák. Je zde velice dobré zabezpečení, kdy kód, který je vložený ve vysílači zabraňuje připojeni jiných vysílačů.

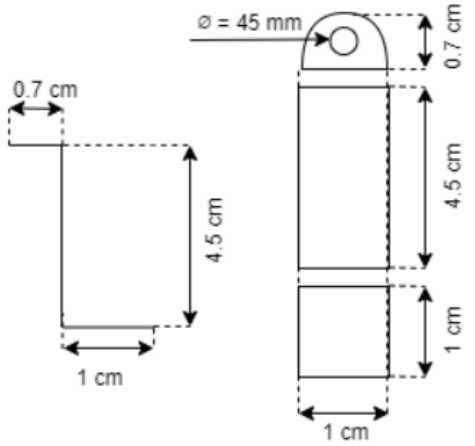
6.4 Vytvoření podkladů pro uchycení komponentů.

Podklady jsou vytvořeny z lehkých a tenkých materiálů tak, aby nezatěžovaly přebytečně samotnou kostru robota.

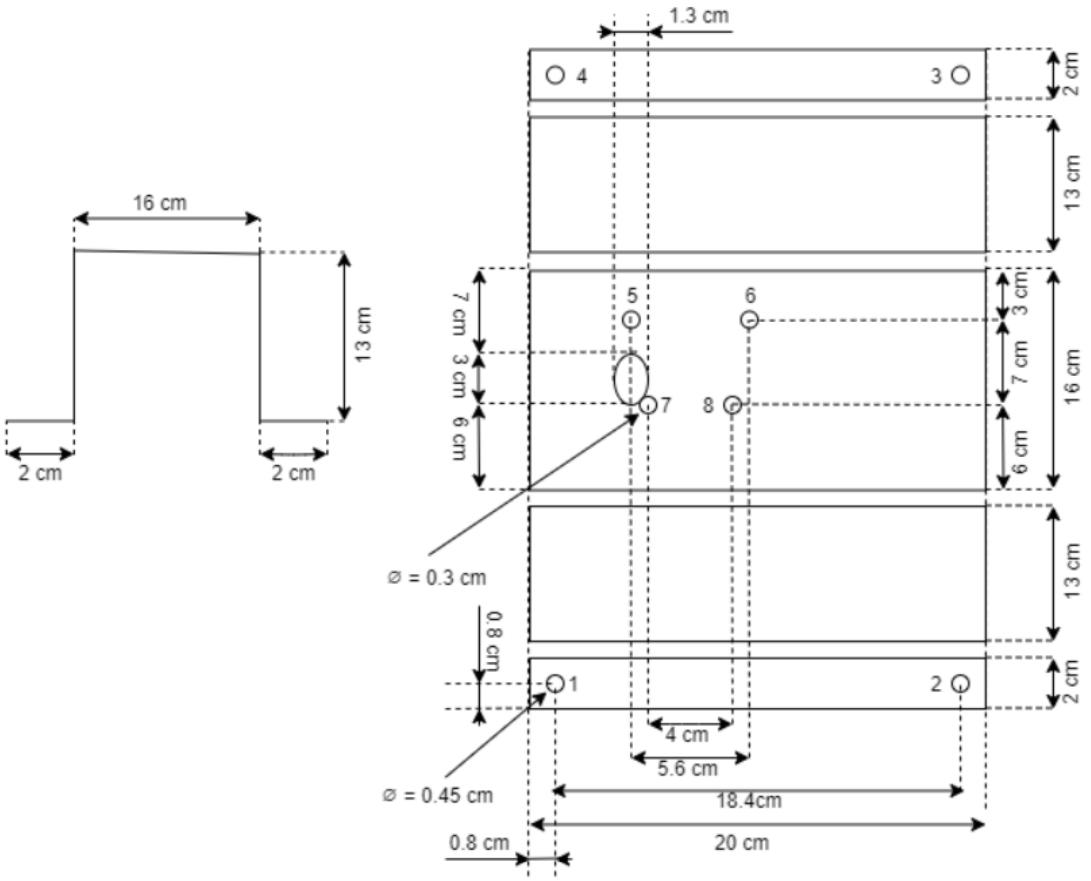


Obrázek 26: Rozměry hlavní podkladové desky

Díry s označením 1 až 10 na hlavní podkladové desce mají průměr 45 mm. Průměr děr 11 a 12 činí u každé z nich 1 cm.



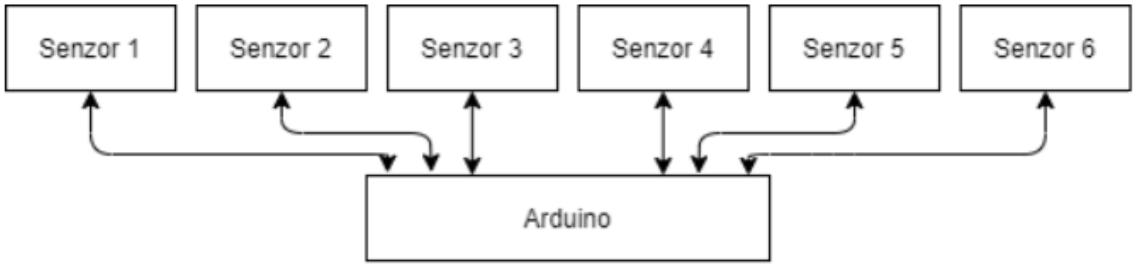
Obrázek 27: Plechový úchyt pro boční senzory – rozměry.



Obrázek 28: Plechový kryt – rozměry

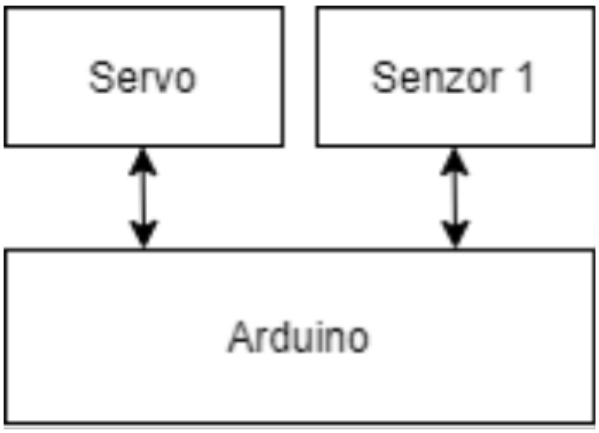
Úchytové díry s označením 1, 2, 3, 4 na krytu mají průměr 0.45 cm. Průměr děr 5, 6, 7, 8 činí u každé z nich 0,3 cm. Tento kryt nám slouží pro uchycení lidaru v nevyšší poloze pro jeho optimální měření.

6.5 Blokové schéma zapojení



Obrázek 29: Blokové schéma zapojení HC-SR04 s Arduinem.

Dané využité senzory jsou zapojeny následovně. Každý z těchto senzorů je připojen na napájení 5 V, které je v blokové schématu zaznačeno červenou barvou. Na desku Arduino jsou dále od senzoru přivedeny kontakty pro TRIG (Oranžová) a ECHO (šedá).



Obrázek 30: Blokové schéma zapojení Arduino radaru.

6.6 Programovací část.

V programu ARDUINO jsem vytvořil kód, pro zaznamenávání hodnot ze senzorů HC-SR04, které jsou ve větším počtu umístěny na mobilním robotu. Ukázku mého navrženého kódu pro jeden senzor naleznete na obrázku 31. Pro zbylé senzory byl kód totožný, měnily se pouze jednotlivá přiřazení u trigPin, echoPin, duration, distance a sensory.

Popis daného kódu: Nejprve jsem si nadefinoval, kam na desce Arduino připojím jednotlivý senzor. V části void setup je nejprve nadefinována komunikace a její rychlost. Následně je stanoveno, který z daných pinů je vstupní a výstupní. V další části je pro pin definována low a high (příkaz pro měření) a jim přiřazena doba měření v mikrosekundách poté je tato hodnota snížená zpětně na low a po uplynutí 2 mikrosekund probíhá následně další měření. Je přečtena délka impulzu, která se převádí na centimetry. Příklad tohoto převodu naleznete v kapitole 5.3. Dále pak následuje podmínka, že pokud je distance1 větší nebo rovno 450 a menší nebo rovno nule vypíše se “Out of range“. Na závěr je již pouze výpis z konzole.

```
int trigPin1=2;
int echoPin1=3;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);

}

void loop() {
  long duration1, distance1;
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = (duration1/2) / 58.31;

   if (distance1 >= 450 || distance1 <= 0){
    Serial.println("Out of range");
  }
  else {
    Serial.print ( "Sensor1  ");
    Serial.print ( distance1);
    Serial.println("cm");
  }
  delay(2000);
}
```

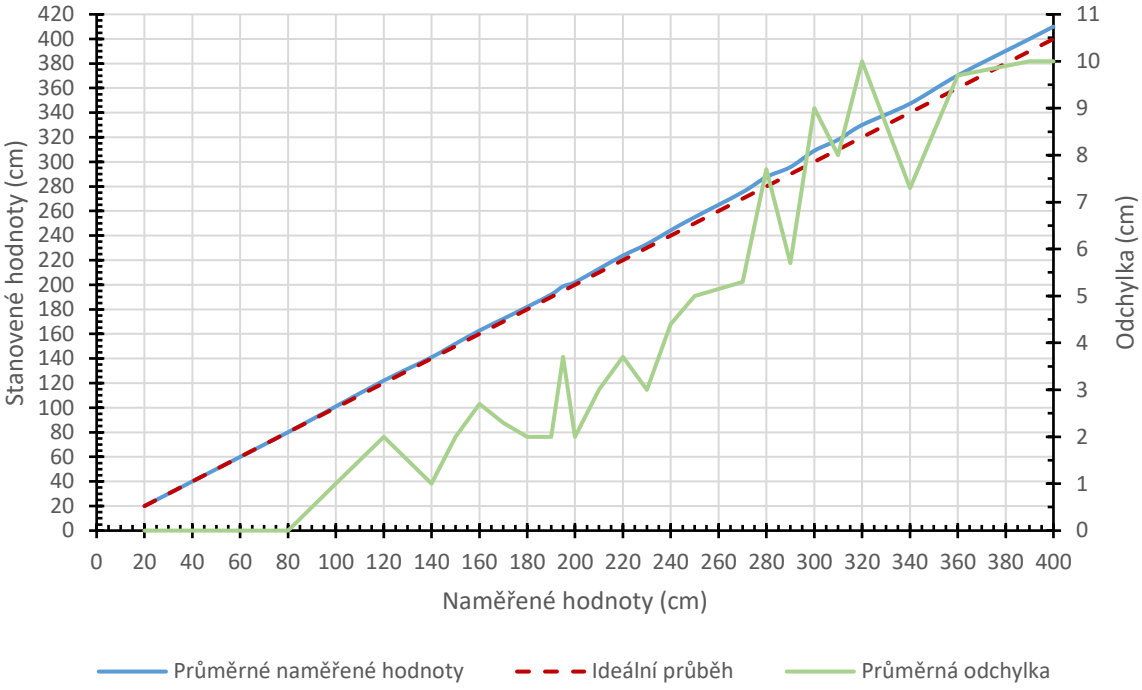
Obrázek 31: Programovací kód pro jeden senzor HC-SR04

6.7 Ověření senzoru pro přesnost měření

V této části jsem ověřoval přesnost měření vzdálenosti senzorů HC-SR04, které jsou umístěny na robotu a slouží pro udání polohy. U daných senzorů výrobce udává měření vzdálenosti od 2 do 400 cm. Pro jednotlivé senzory bylo provedeno ověřovací měření vzdálenosti proti různým povrchům, na které mohou senzory narazit. Každý materiál má různé odrazové vlastnosti. Pro toto ověření byly využity překážky z materiálu: sklo, plech, plast, dřevo a látka. Toto měření bylo prováděno ve velkém prostoru při pokojové teplotě, kde jsem měl jistou, že se ve vyzařovaném pásmu tohoto senzoru nenachází žádná překážka, která by měření ovlivnila. Jednotlivé materiály sklo a látka byly měřeny s rozměry 100 x 100 cm, plast a dřevo o rozměrech 60 x 60 cm a plech o velikosti 30 x 30 cm. Pro dosažení nejmenší nepřesnosti byl využitý senzor HC – SR04 uchycen do nepájivého pole, přes které byl propojen s deskou Arduino UNO R3 a ta byla následně propojena s počítačem. Na podlahu daného prostoru jsem uchytil za pomocí lepící pásky metr a požádal druhou osobu o přikládání překážky na mnou požadované hodnoty. Pro každý materiál byly provedeny tři měření a následně byly hodnoty zprůměrovány a byla určena odchylka jednotlivých měření. Veškerá data naleznete v tabulkách 6 až 10. Na následujících grafech naleznete zprůměrované měření s danými materiály a průměrnou odchylku měření, které bylo prováděno ve volném prostoru za pomocí senzorů, počítače, metru a jednotlivých materiálu.

Tabulka 6: Naměřené hodnoty od senzoru k skleněné překážce.

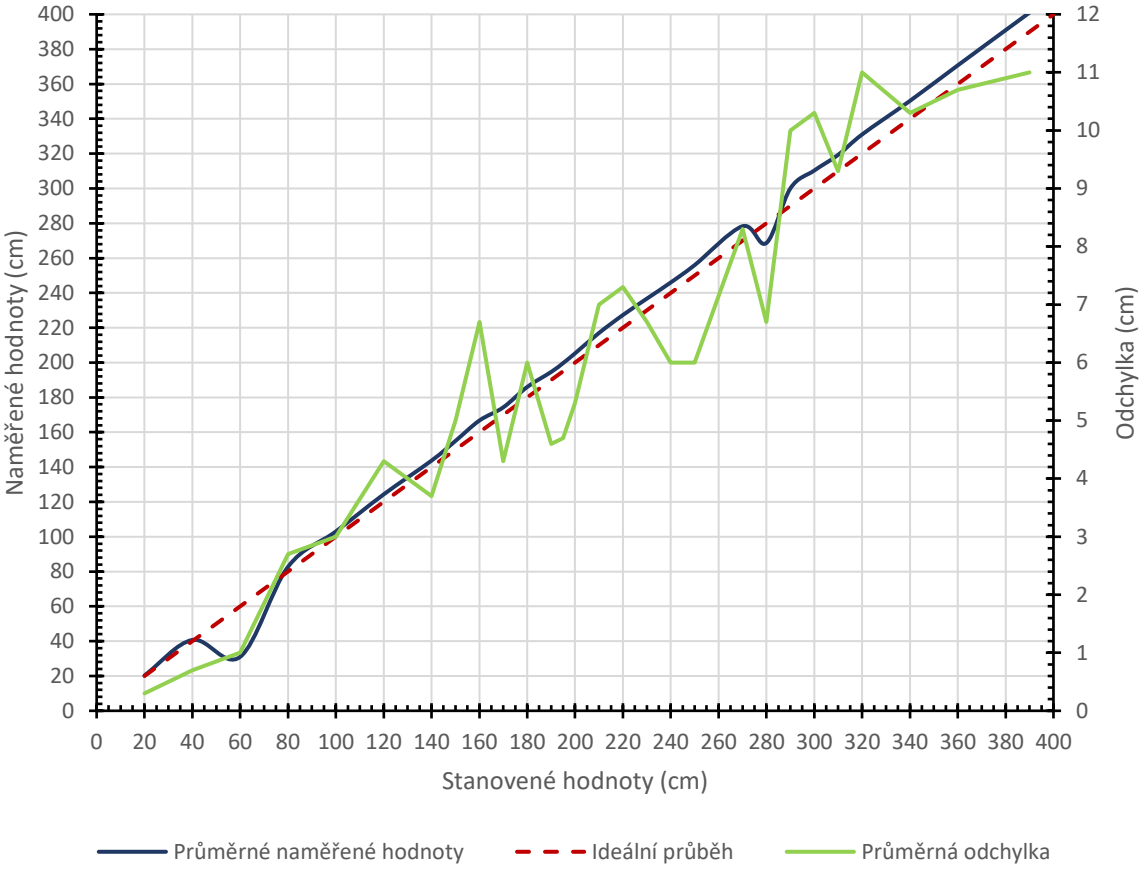
Stanovené hodnoty (cm)	Naměřené hodnoty 1 (cm)	Naměřené hodnoty 2 (cm)	Naměřené hodnoty 3 (cm)	Průměrné naměřené hodnoty (cm)	Průměrná odchylka (cm)
20	20	20	20	20	0
40	40	40	40	40	0
60	60	60	60	60	0
80	80	80	80	80	0
100	101	101	101	101	1
120	122	122	122	122	2
140	141	141	141	141	1
150	152	152	152	152	2
160	162	162	163	162,7	2,7
170	171	173	173	172,3	2,3
180	182	182	182	182	2
190	191	193	192	192	2
195	197	199	200	198,7	3,7
200	201	203	202	202	2
210	212	214	213	213	3
220	224	224	223	223,7	3,7
230	233	232	234	233	3
240	245	243	245	244,3	4,4
250	256	254	255	255	5
270	274	276	276	275,3	5,3
280	288	287	288	287,7	7,7
290	296	295	296	295,7	5,7
300	308	310	309	309	9
310	317	319	318	318	8
320	329	331	330	330	10
340	345	349	348	347,3	7,3
360	368	372	371	370,3	9,7
390	400	399	401	400	10
400	410	-	-	410	10



Obrázek 32: Graf znázorňující chyby měření vzdálenosti proti sklu

Tabulka 7: Naměřené hodnoty od senzoru k plechové překážce.

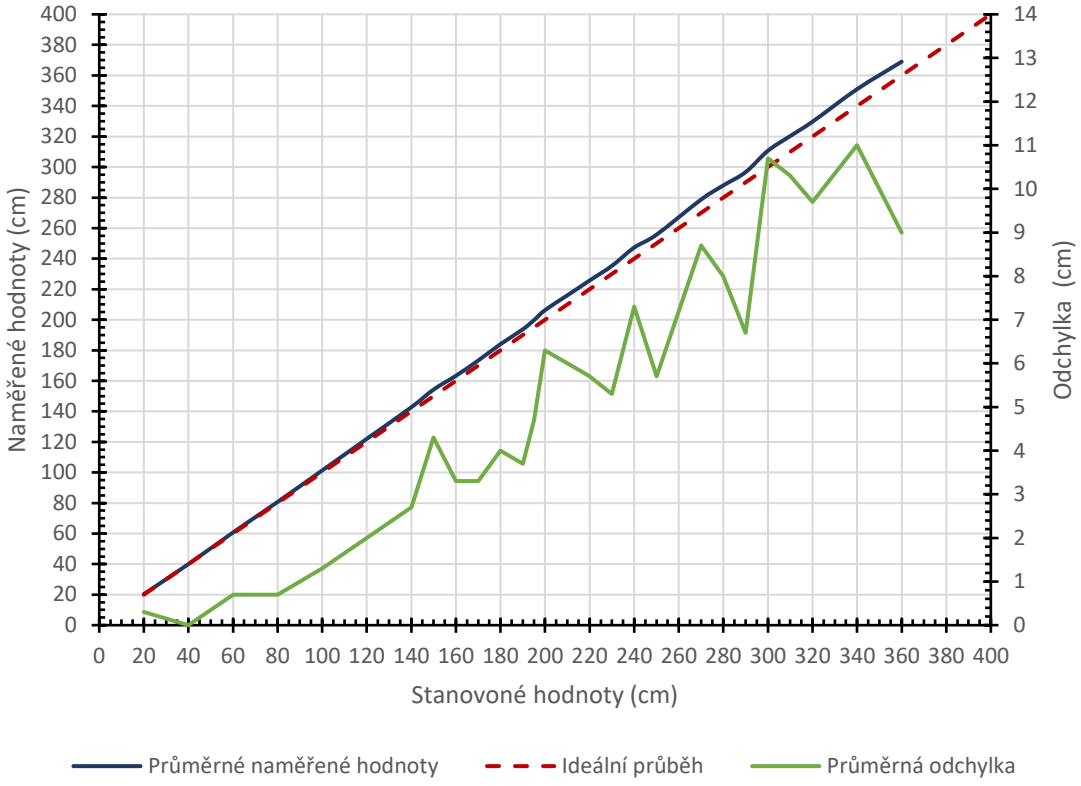
Stanovené hodnoty (cm)	Naměřené hodnoty 1 (cm)	Naměřené hodnoty 2 (cm)	Naměřené hodnoty 3 (cm)	Průměrné naměřené hodnoty (cm)	Průměrná odchylka (cm)
20	21	20	20	20,3	0,3
40	42	40	40	40,7	0,7
60	62	61	60	31	1
80	84	82	82	82,7	2,7
100	105	103	102	103	3
120	124	124	125	124,3	4,3
140	143	144	144	143,7	3,7
150	155	155	155	155	5
160	167	167	166	166,7	6,7
170	174	173	176	174,3	4,3
180	187	185	186	186	6
190	196	194	194	194,6	4,6
195	201	199	199	199,7	4,7
200	206	204	206	205,3	5,3
210	217	217	217	217	7
220	228	226	228	227,3	7,3
230	236	237	237	236,7	6,7
240	247	245	246	246	6
250	257	254	257	256	6
270	277	279	279	278,3	8,3
280	287	286	287	286,7	6,7
290	300	301	299	300	10
300	311	310	310	310,3	10,3
310	321	319	318	319,3	9,3
320	332	331	330	331	11
340	350	351	350	350,3	10,3
360	372	370	370	370,7	10,7
390	403	399	-	401	11
400	-	-	-	-	-



Obrázek 33: Graf znázorňující chyby měření vzdálenosti proti plechu.

Tabulka 8: Naměřené hodnoty od senzoru k plastové překážce.

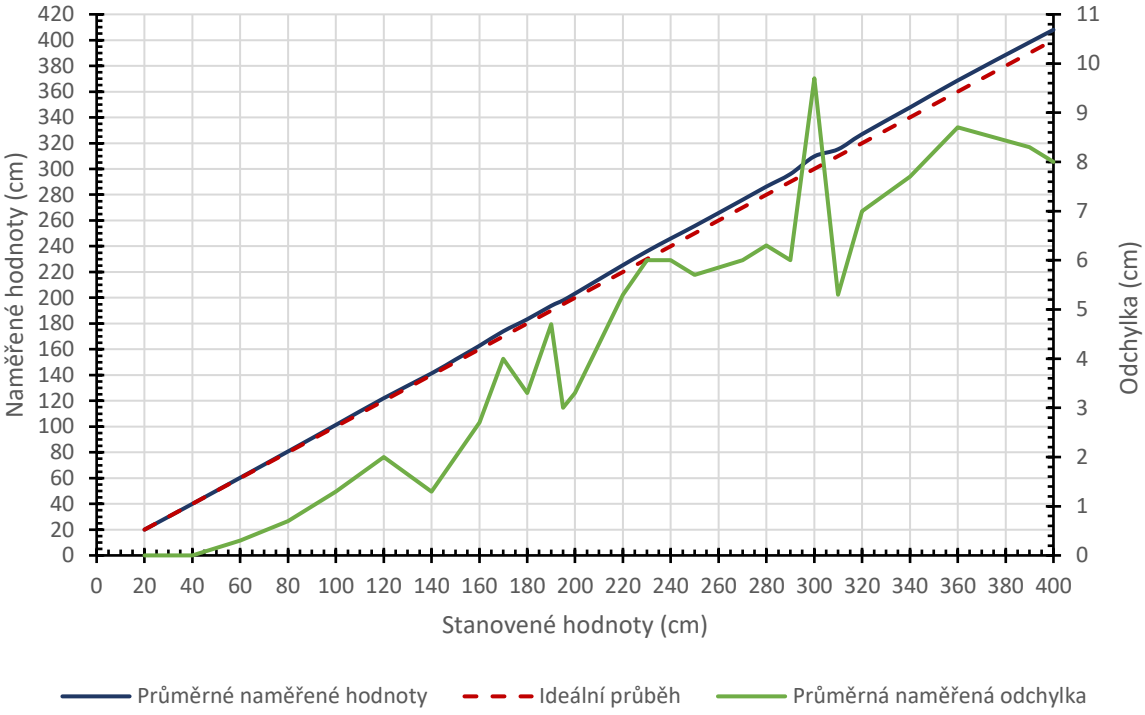
Stanovené hodnoty (cm)	Naměřené hodnoty 1 (cm)	Naměřené hodnoty 2 (cm)	Naměřené hodnoty 3 (cm)	Průměrné naměřené hodnoty (cm)	Průměrná odchylka (cm)
20	21	20	20	20,3	0,3
40	40	40	40	40	0
60	61	60	61	60,7	0,7
80	81	80	81	80,7	0,7
100	102	101	101	101,3	1,3
120	122	121	123	122	2
140	143	142	143	142,7	2,7
150	155	155	153	154,3	4,3
160	163	164	163	163,3	3,3
170	174	174	172	173,3	3,3
180	184	185	183	184	4
190	194	193	194	193,7	3,7
195	200	199	200	199,7	4,7
200	205	207	207	206,3	6,3
210	217	216	215	216	6
220	225	226	226	225,7	5,7
230	236	236	234	235,3	5,3
240	246	248	248	247,3	7,3
250	256	255	256	255,7	5,7
270	279	278	279	278,7	8,7
280	287	289	288	288	8
290	297	297	296	296,7	6,7
300	310	310	312	310,7	10,7
310	320	318	323	320,3	10,3
320	330	328	331	329,7	9,7
340	353	349	351	351	11
360	-	-	369	369	9
390	-	-	-	-	-
400	-	-	-	-	-



Obrázek 34: Graf znázorňující chyby měření vzdálenosti proti plastu.

Tabulka 9: Naměřené hodnoty od senzoru k dřevěné překážce.

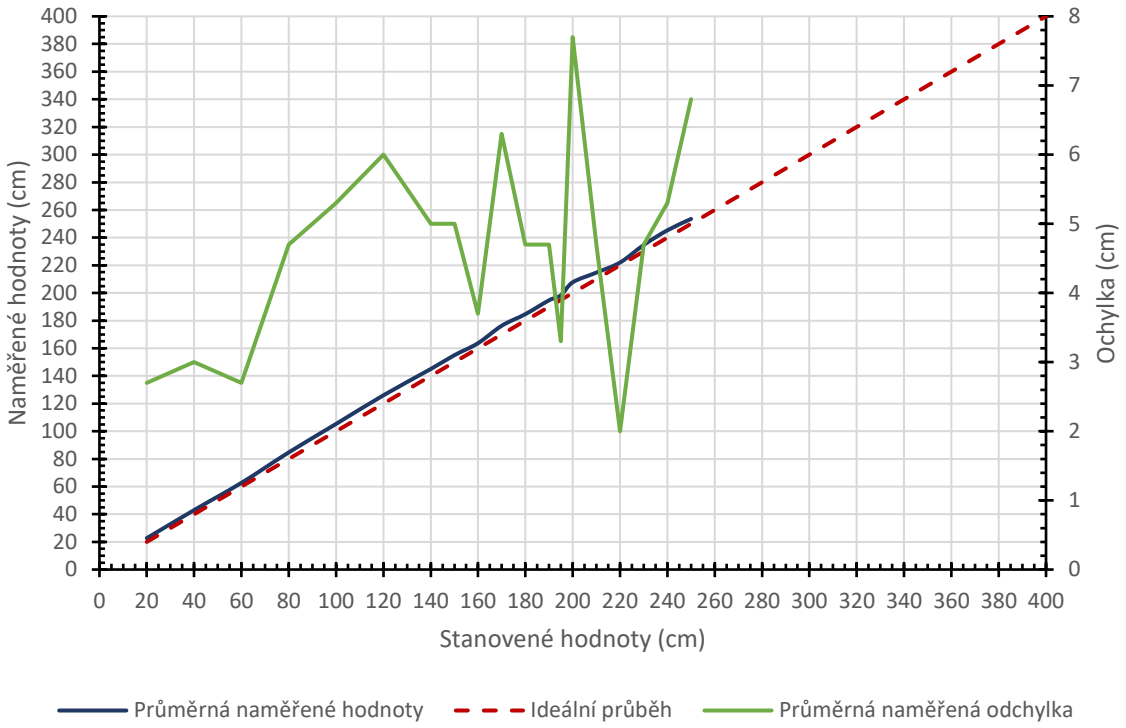
Stanovené hodnoty (cm)	Naměřené hodnoty 1 (cm)	Naměřené hodnoty 2 (cm)	Naměřené hodnoty 3 (cm)	Průměrné naměřené hodnoty (cm)	Průměrná odchylka (cm)
20	20	20	20	20	0
40	40	40	40	40	0
60	60	60	61	60,3	0,3
80	81	80	81	80,7	0,7
100	102	101	101	101,3	1,3
120	122	123	121	122	2
140	141	141	142	141,3	1,3
150	152	152	152	152	2
160	163	162	163	162,7	2,7
170	174	173	175	174	4
180	185	182	183	183,3	3,3
190	194	195	192	193,7	4,7
195	197	199	198	198	´3
200	204	202	204	203,3	3,3
210	215	213	215	214,3	4,3
220	226	225	225	225,3	5,3
230	237	235	236	236	6
240	247	245	246	246	6
250	257	254	256	255,7	5,7
270	277	275	276	276	6
280	287	286	286	286,3	6,3
290	297	296	295	296	6
300	309	309	311	309,7	9,7
310	316	314	316	315,3	5,3
320	328	327	326	327	7
340	348	348	347	347,7	7,7
360	369	369	368	368,7	8,7
390	399	397	399	398,3	8,3
400	411	-	405	-	8



Obrázek 35: Graf znázorňující chyby měření vzdálenosti proti dřevu.

Tabulka 10: Naměřené hodnoty od senzoru k látkové překážce.

Stanovené hodnoty (cm)	Naměřené hodnoty 1 (cm)	Naměřené hodnoty 2 (cm)	Naměřené hodnoty 3 (cm)	Průměrné naměřené hodnoty (cm)	Průměrná odchylka (cm)
20	24	22	22	22,7	2,7
40	44	43	42	43	3
60	62	63	63	62,7	2,7
80	85	83	86	84,7	4,7
100	105	104	107	105,3	5,3
120	125	128	125	126	6
140	147	143	145	145	5
150	155	154	156	155	5
160	163	163	165	163,7	3,7
170	178	175	176	176,3	6,3
180	184	187	183	184,7	4,7
190	194	194	196	194,7	4,7
195	197	197	201	198,3	3,3
200	206	209	208	207,7	7,7
210	213	215	216	214,7	4,7
220	219	222	225	222	2
230	233	236	235	234,7	4,7
240	244	244	248	245,3	5,3
250	248	-	259	253,5	6,8
270	-	-	-	-	-
280	-	-	-	-	-
290	-	-	-	-	-
300	-	-	-	-	-
310	-	-	-	-	-
320	-	-	-	-	-
340	-	-	-	-	-
360	-	-	-	-	-
390	-	-	-	-	-
400	-	-	-	-	-



Obrázek 36: Graf znázorňující chyby měření vzdálenosti proti látce.

6.8 Vyhodnocení měření ultrazvukovým senzorem

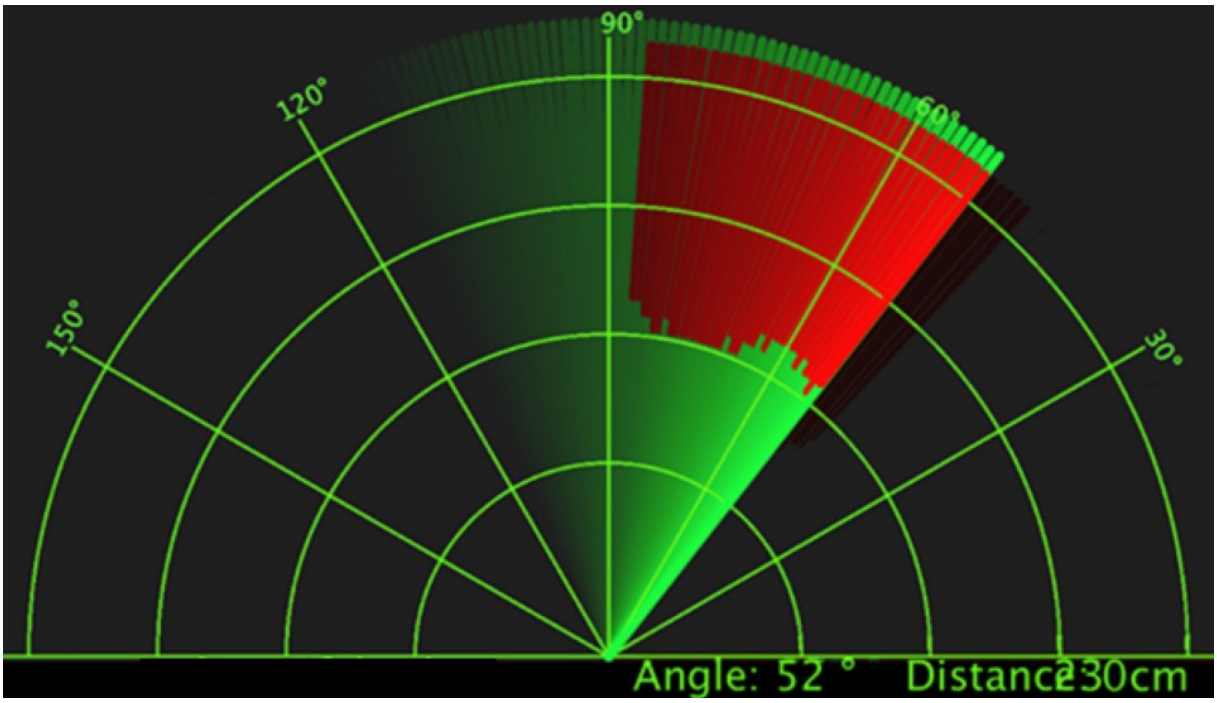
Výrobce udává, že dané senzory měří s velkou přesností od 2 do 200 cm, poté se přesnost měření zhoršuje až do 400 cm. Ověřování měření přesnosti daného senzoru bylo prováděno za stejných podmínek pro veškeré materiály, přesto jsou odchylky některých měření poměrně velké. Tyto velké odchylky jsou způsobeny drsností povrchu daných materiálu, kterou musíme zohledňovat. Následně musíme vzít v potaz, že jednotlivé materiály o různých rozměrech byly na mnou požadované vzdálenosti od senzorů umisťovány druhou osobou.

Tento způsob zapříčinil ovlivnění měření, a to tím způsobem, že daný předmět nebyl ve stejném úhlu na všech měřených hodnotách a každé jeho natočení i řádově zlomky působí na odraženou energii. Pokud bychom chtěli dosáhnout přesného měření s minimálními odchylkami je potřeba například kolejnice na jejímž konci je upevněn úchyt, do kterého se daný materiál pevně uchytí, aby se neměnilo natočení odrazného předmětu vůči přijímači a vysílači.

6.9 Servomotor a ultrazvukový senzor

Pro toto měření bylo využito desky Arduino, servomotoru a senzoru HC-SR04. Daný senzor se uchytí na servomotor a ten se následně propojí s deskou Arduino, ve které byl napsán kód pro pohon servomotoru a získávání hodnot ze senzoru. Tyto hodnoty byly následně převedeny do programu Processing, ve kterém jsem vytvořil program umožňující dané vykreslování v reálném čase, které lze vidět na obrázku 37.

Měření touto metodou je problematické, protože natočení servomechanismu kmitá. Je zde zpětnovazební potenciometr, kdy šum na potenciometru může se servomotorem kmitat nebo jej nedostane do všech úhlu. Tento problém nastal i zde kde daný servomotor nedokázal zaměřit hodnoty v maximálním natočení a snímal překážky v natáčecím úhlu od 0° do 180°.

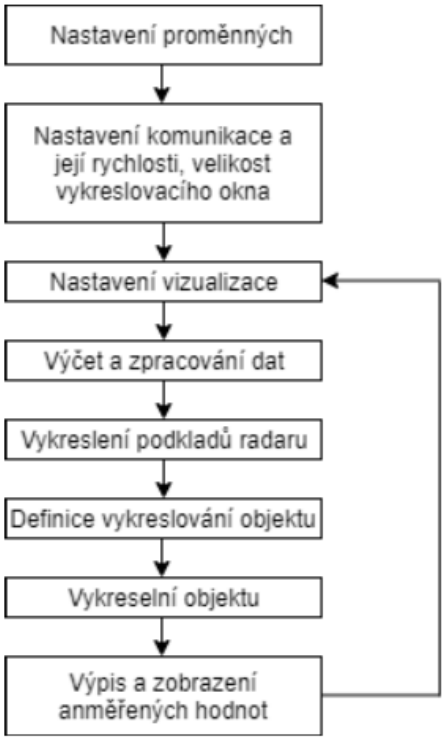


Obrázek 37: Měření vzdálenosti za pomocí servomotoru a ultrazvuku

Na daném obrázku je červeně znázorněna překážka, která se vyskytla před robotem. Z obrázku je patrné, v jakém úhlu a vzdálenosti se daná překážka nachází.

```
#include <Servo.h>.
const int trigPin = 10;
const int echoPin = 11;
long duration;
int distance;
Servo myServo;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  myServo.attach(12);
}
void loop() {
  for(int i=0;i<=180;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
  for(int i=180;i>0;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= (duration1/2)/58.31;
  return distance;
}
```

Obrázek 38: Vytvořený kód pro funkci Arduina.



Obrázek 39: Blokové schéma programu Processing.

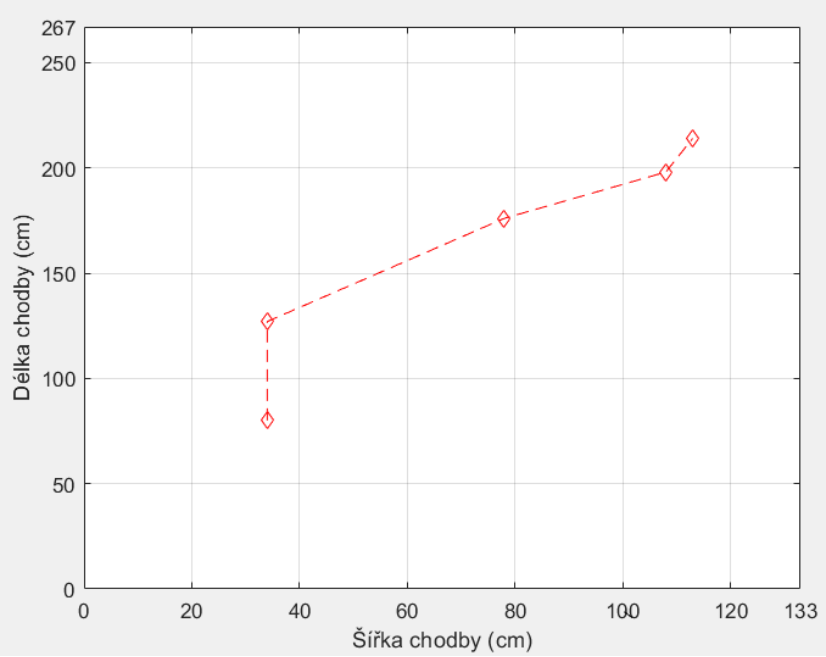
6.10 Získána data a měření polohy robota

Data pro měření polohy mobilního robota byly získávány ze senzorů HC-SR04. Tyto data naleznete v tabulce 11. Data byly následně v reálném čase převáděný do programu Excel, kde je potřeba jednotlivá data upravit, protože senzor neměřil přesnou polohu středu robota. Bylo nutné k změřeným hodnotám ze senzoru 4 připočíst dalších 17 cm a k senzoru 6 přičíst hodnotu 29 cm, které chybí do středu robota od umístěného senzoru.

Mým hlavním a využitým řešením bylo v reálném čase převedení dat ze senzorů do programu Matlab, který nám vykresli trasu, kudy se robot zhruba pohyboval a bodově vyznačí místa měření polohy, kde se robot zastavil viz. obrázek 40. Toto měření probíhalo v místnosti 133 x 268 cm. Celá tato místnost byla obložena dřevěným obkladem a nebylo zde nic, co by ovlivnilo odrazy měření jednotlivých senzorů. Měření polohy robotu probíhalo vždy za klidového stavu při stejné pokojové teplotě. Veškeré součástky jsou pevně uchyceny a nic nenarušit jejich měření. Výsledné hodnoty poloh změřené za pomocí senzorů 2, 4, 5, 6 nám krásně zobrazují, jak přesné jsou senzory za klidového stavu.

Tabulka 11: Hodnoty získané ze senzorů v centimetrech.

Měření	Senzor 2	Senzor 4	Senzor 5	Senzor 6
1	158	17	82	51
2	111	17	82	98
3	62	61	38	147
4	40	91	8	169
5	24	96	3	185



Obrázek 40: Vykreslení naměřených pozic robota.

Následně pro měření polohy v reálném čase byly získávána data z Lidaru. Tyto data jsou uvedené pro každý jednotlivý úhel v okruhu 360°.

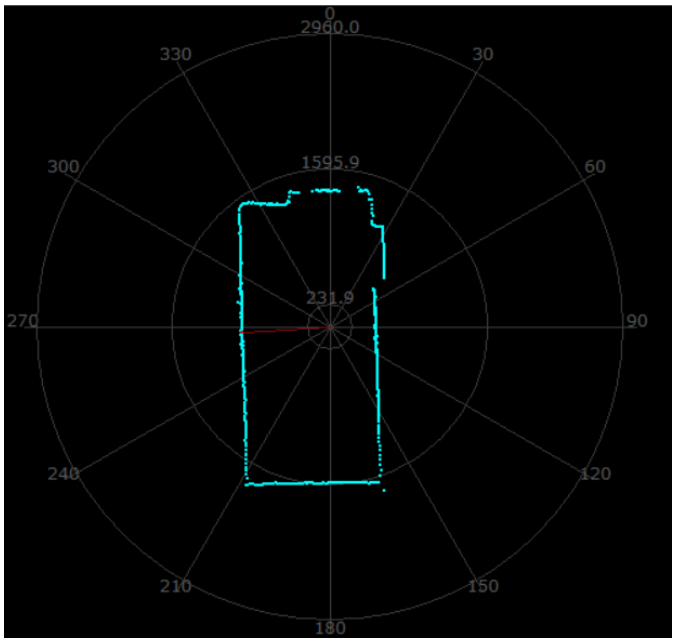
Tabulka 12: Hodnoty získané z lidaru.

Úhel (°)	Vzdálenost (mm)
10	2074
30	3113
76	9977
153	5779
190	4205
278	5583
350	2568

6.11 Lidarové vykreslení prostoru a určení polohy

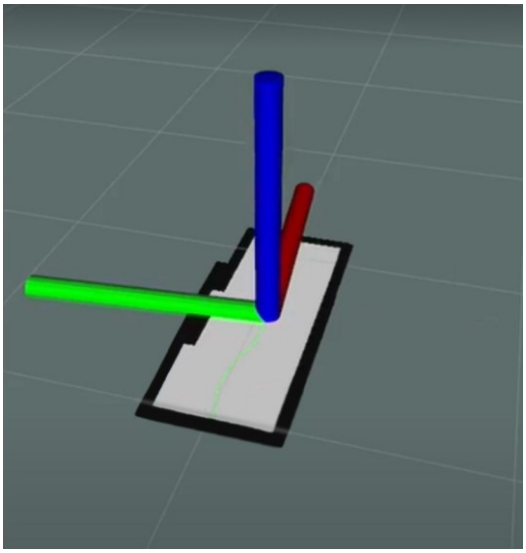
Za pomocí RPLIDARU A1 bylo mým cílem vykreslit místnost, kde se daný robot pohyboval a určit jeho polohu. Toto vykreslování se vytváří za pomocí aplikace, kterou poskytuje společnost Slamtec. Během mého pokusu o měření se vyskytl problém, který daná společnost nebyla schopná opravit. Jednalo se o nemožnost stažení potřebného programu pro vykreslování. Byl jsem tedy nucen použit program framegrabber Module od společnost Slamtec, který má velké omezení funkcí.

Za pomocí využití náhradního programu bylo možno vykreslit danou místnost. Na obrázku 41 lze vidět, že se robot vždy nachází ve středu kruhu, kde se v dané místnosti pohybuje. Následně můžeme také změřit jeho polohu za pomocí metody, která je popsaná v kapitole 6.6. Na obrázku je vidět červená čára, která udává hodnotu vzdálenosti od jednotlivých překážek a stěn v místnosti.



Obrázek 41: Lidarové vykreslení prostoru.

Následně bylo prováděno vykreslování místnosti, měření polohy a zaznamenávání trasy robota za pomocí framework ROS. Uvedení ROSu do provozu naleznete v kapitole 5.5. ROS byl zprovozněn na operačním systému Linux, který mi následně umožnovat udělat fúzi dat ze senzorů a vytvořit potřebnou komunikaci. Data ze senzorů jsem ukládal do lokální databáze NOTE, která je součástí ROSu a umožnuje také sdružovat data z odometrie. Následně bylo možno vytvořit prostředí pro vykreslování a určování polohy robotu. Vykreslování jsem prováděl pomocí Hector Slam metody 6.1.



Obrázek 42: Lidarové 2D vykreslení prostoru.

Za pomocí použití Raspberry Pi 3 máme možnost dálkové komunikace například do mobilního telefonu, tabletu nebo jiného pc. V těchto zařízeních si můžeme vyobrazovat vykreslování prostoru pro určení polohy a měření aktuální polohy. Toto dálkové propojení bylo provedeno za pomocí následujícího postupu. V prvním kroku se nastavil VNC servr v Raspberry Pi 3. Raspberry bylo připojeno do lokální sítě LAN (Local Area Network). Po přidělení IP adresy pro Raspberry tuto adresu zadáme do programu VNC viewer. Následně se v jiném dostupném zařízení provede instalace VNC viewer, kde se do programu zadává IP adresa a navazuje se spojení s Raspberry. Jednotka Raspberry si pro dané spojení vyžádá přihlašovací údaje (Přihlašovací jméno: PI, Heslo: 0000). Po přihlášení se nám vyobrazí prostředí operačního systému Raspberry. V tomto rozhraní nadále ovládáme Raspberry. Daným způsobem si můžeme zobrazovat vykreslování z lidaru a výpis hodnot ze senzorů pro měření vzdálenosti.

V programu Python byl napsán program, který umožnuje vykreslování prostoru a zobrazení aktuální polohy, které vidíme na obrázku 44.

Program nepřetržitě a neustále čte a vykresluje data do jeho zastavení.

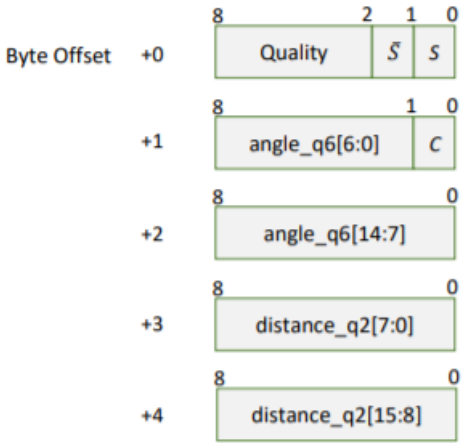
```
try:
    for scan in lidar_scans(lidar):
      for (_, angle, distance) in scan:
        scan_data[min([359, floor(angle)])] = distance
      process_data(scan_data)


except KeyboardInterrupt:
   print('Stoping.')
lidar.stop()
lidar.disconnect()
```

Pokud chceme přečíst data z lidaru je zapotřebí poslat SCANE_BYTE, který řekne lidaru, aby spustil skenování.

```
self._send_cmd(b'\x20')
raw = self._read_response(dsize)
```

Data přečtena z lidaru za pomocí funkce _read_responze o délce pěti bytů. V těchto 5 bytech jsou obsažená data z lidaru – vzdálenost, úhel a kvalita.

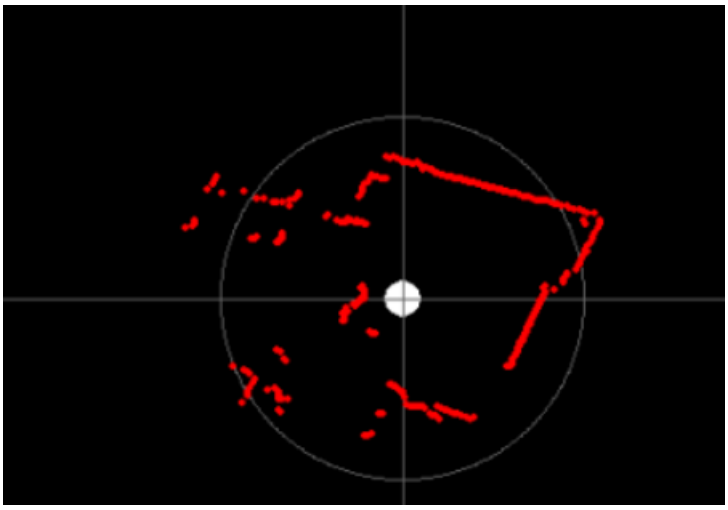


Obrázek 43: Formát rozložení dat z lidaru. [37]

```
angle = ((raw[1] >> 1) + (raw[2] << 7)).
distance = (raw[3] + (raw[4] << 8)).
quality = raw[0] >> 2
```

První byte obsahuje kvalitu, druhý a třetí obsahují úhel a čtvrtý a pátý obsahují vzdálenost. Z těchto bytu jsme schopni získat informace o daném stavu. Je potřeba posunout první byte o dva bity

doprava a tím získáme výslednou kvalitu. Následně se úhel získává tím, že druhý byte posuneme o jeden bit doprava a sečteme se třetím bytem posunutým o 8 bitů do leva. Vzdálenost získáme sečtením čtvrtého a pátého bytu posunutého o 8 bitu doleva. Poté se vykreslují body podle vzdálenosti a úhlů.



Obrázek 44: Lidarové vykreslení za pomocí Raspberry Pi 3.

Pro měření aktuální polohy za pomocí ultrazvukových senzoru byla k Raspberry Pi 3 připojena deska Arduino a nainstalován ovládací program. V tomto prostředí se využil kód pro ultrazvukové senzory, který je popsán v části 6.6. Z tohoto kódu získáváme data o poloze, která jsou na příkladu uvedena v tabulce 11.

7 Závěr

V teoretické části bakalářské práce jsem popsal metody lokalizace, měření a vykreslování za pomocí SLAM metody. Dále byly popsány snímače, které je možno využít pro měření polohy a následně využité snímače pro měření.

V praktické části bylo navrženo vlastní řešení pro měření polohy mobilního robota v budově nebo místnosti. Pro měření a určení polohy robota byly využity ultrazvukové senzory a RPLidar A1. Nejprve byl popsán použitý robot, následně pohon samotného robota a vytvořeny podklady pro možnosti uchycení jednotlivých senzoru. Dalším krokem je popsání jednotlivých zapojení využitých senzorů a jejích zprovoznění v prostředí od Arduina, Processingu a Pythonu. V kapitole 6.6 je podrobně popsaná programovací část pro ultrazvukové senzory. Provedl jsem ověřování přesnosti ultrazvukových senzoru. Toto ověření je zhodnoceno v kapitole 6.8. V následujícím kroku bylo vytvořeno měření polohy robota a vzdálenosti od překážek za pomocí ultrazvukového senzoru uchyceného na servomotoru, který je popsán v kapitole 6.9. Tato kapitola obsahuje kód vytvořený v Arduinu a blokové schéma kódu vytvořeného v programu Processing. V kapitole 6.10 je popsáno měření polohy mobilního robota za pomocí několika ultrazvukových senzorů, jejichž hodnoty se v reálném čase převádějí do programu Matlab, kde probíhá vykreslování v reálném čase nebo do Excelu. Data o poloze v reálná čase robota jsem získával také z Lidaru, který nám vypisuje data v okruhu $360°$.

V kapitole 6.11 bylo prováděno měření a vykreslování polohy mobilního robota za pomocí ROSu a SLAM metody a využitých senzorů pro měření – Lidaru a ultrazvukových senzorů. V této části byly dané senzory také napojeny na Raspberry Pi 3, které nám umožnuje vzdálenou komunikaci. Rovněž jsem popsal funkčnost daného programu, funkce vzdáleného ovládání, vykreslování a zobrazování dat pro měření polohy.

Na závěr lze konstatovat, že zadání bakalářské práce bylo splněno za pomocí ultrazvukových senzorů, Lidaru, servomotoru s ultrazvukem.

8 Literatura

[1] GREŠ, Tomáš. ATESYSTEM. Strojové vidění a kamery. Ostrava, 2016. Materiál společnosti ATEsystem, s.r.o. Dostupné z: http://kamery.atesystem.cz/know-how/strojove-videni/

[2] Ultrazvukový měřič vzdálenosti: HC-SR04 [online]. Havlíčkův Brod: Arduino-shop, 2016 [cit. 2020-12-19]. Dostupné z: https://arduino-shop.cz/arduino/846-eses-ultrazvukovy-meric-vzdalenosti-hc-04-pro-jednodeskove-pocitace.html

[3] Ultrazvukový měřič vzdálenosti: HC-SR04 [online]. Havlíčkův Brod: Luboš M, 2016 [cit. 2020-12-19]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/meric-vzdalenosti-ultrazvukovy.html

[4] Laserový skener: RPLIDAR A1 [online]. Havlíčkův Brod: Arduino-shop, 2009 [cit. 2020-12-19]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/941/rplidar_a1m8_datasheet.pdf

[5] Trilaterace [online]. Web: commons.wikimedia, 2020 [cit. 2020-12-19]. Dostupné z: https://commons.wikimedia.org/wiki/File:Trilateration.png

[6] GREŠ, Tomáš. ATESYSTEM. Strojové vidění a kamery. Ostrava, 2016. Materiál společnosti ATEsystem, s.r.o. Dostupné z: http://kamery.atesystem.cz/know-how/strojove-videni/

[7] BOWDITCH, Nathaniel. The American Practical Navigator : An Epitome of Navigation [online]. 1995. Bethesda, Maryland : National Imagery and Mapping Agency, 1995 [cit. 2020-19-12]. Dead Reckoning, s. 113–118. URL: <http://www.irbs.com/bowditch/>.

[8] Odometry and Dead Reckoning, 2006 Dostupné z: http://www.rst.e-technik.uni-dortmund.de/cms/Medienpool/Downloads/Lehre/Vorlesungen/Robotics_Theory/

[9] Optické senzory [online]. Praha: eatonelektrotechnika, 2015 [cit. 2020-12-19]. Dostupné z: http://www.eatonelektrotechnika.cz/opticke-senzory.html

[10] EVERETT, H. R. 1995. Sensors for mobile robots: theory and application. Wellesley, Mass.: A.K. Peters, 528 p. ISBN 1-56881-048-2.

[11] True-range multilateration [online]. Wikimedia: Wikimedia, 2020 [cit. 2020-12-19]. Dostupné z: https://en.wikipedia.org/wiki/True-range_multilateration

[12] True-range multilateration [online]. Robotika.cz: Robotika.cz, 2005 [cit. 2020-12-19]. Dostupné z: https://robotika.cz/guide/odometry/cs

[13] NOVÁK, P. 2007. Mobilní roboty - pohony, senzory, řízení, BEN-technická literatura, Praha, ISBN 80-7300-141-1

[14] Detektor barvy TCS230 [online]. Havlíčkův Brod: ECLIPSERA, 2016 [cit. 2020-12-19]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/arduino-detektor-barvy.html

[15] RIPKA, Pavel a Alois TIPEK, ed. Modern sensors handbook. Hoboken: John Wiley & Sons, 2010. Instrumentation and measurement series. ISBN 978-0-470-61223-1.

[16]FRADEN, Jacob. Handbook of modern sensors: physics, designs, and applications. 3rd ed. New York: Springer, c2004. ISBN 0-387-00750-4.
[17]Technologie radaru: Princip a technologie radaru oblasti použití význam pro GIS [online]. In: . s. 5-6 [cit. 2021-03-13]. Dostupné z: http://wiki.cs.vsb.cz/images/4/4b/Lic060-gis-radar.pdf
[18]Princip radaru [online]. , 1 [cit. 2021-03-14]. Dostupné z: https://www.army.cz/images/id_8001_9000/8753/radar/k23.htm
[19]ARDUINO. Bastlirna.hwkitchen.cz [online]. 2014 [cit. 2021-03-14]. Dostupné z: https://bastlirna.hwkitchen.cz/programujeme-arduino/
[20]GIANCOLA, Silvio, Matteo VALENTI a Remo SALA. A survey on 3d cameras: metrological comparison of time-of-flight, structured-light and active stereoscopy technologies [online]. New York, NY: Springer Science Business Media, 2018 [[cit. 2021-03-14].
[21]Aloysius Wehr, Uwe Lohr, Airborne laser scanning—an introduction and overview, ISPRS Journal of Photogrammetry and Remote Sensing, 1999, ISSN 0924-2716, http://knightlab.org/rscc/readings/airborne_laser_scanning.pdf
[22]LI, Larry. Time-of-Flight Camera - An Introduction [online]. In: . 2014 [cit. 2021-03-18]
[23]Prostředky relativní lokalizace [online]. 2009 [cit. 2021-03-18]. Dostupné z: http://marek.sk.sweb.cz/lokalizace/kapitola3.html. Diplomová práce. Univerzita Karlova.
[24]LOKALIZACE POMOCÍ LASEROVÉHO DÁLKOMĚRU SICK [online]. 2010 [cit. 2021-03-18]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=29209. Bakalářská práce. Vysoké učení technické v Brně.
[25]QuicRun WP 880 (80A) pro dva motory. Profimodel.cz [online]. [cit. 2021-03-28]. Dostupné z: https://profimodel.cz/cs/auto-lodni/246565-quicrun-wp-880-80a-pro-dva-motory-6938994413138.html?gclid=CjwKCAjwr_uCBhAFEiwAX8YJgfQYv35_0GobnJnKYQm5qFDUrDTC6d7Z9Yvf8TB_YyJDXJy2G1ZJqRoCt44QAvD_BwE#246565k
[26]MIG 500 Turbo 7,2 V Race. Eckamodel.cz [online]. [cit. 2021-03-28]. Dostupné z: https://www.peckamodel.cz/133078-mig-500-turbo-7-2-v-race
[27]Parametry signálu poskytovaný RC přijímači Spektrum. Astramodel.cz [online]. [cit. 2021-03-28]. Dostupné z: https://www.astramodel.cz/cz/blog/parametry-signalu-poskytovany-rc-prijimaci-spektrum.html
[28]Spektrum přijímač AR400 DSM2/DSMX 4CH Micro. Astramodel.cz [online]. [cit. 2021-03-28]. Dostupné z: https://www.astramodel.cz/cz/katalog/spektrum/spektrum-dsm-x-prijimac-4ch-micro-ar400-p27206.html
[29]Spektrum DX6i DSM X Spektrum Air Heli - AR6210 M1. Tsbohemia.cz [online]. [cit. 2021-03-28]. Dostupné z: https://www.tsbohemia.cz/spektrum-dx6i-dsm-x-spektrum-air-heli-ar6210-m1_d193851.html
[30]Time-of-flight-senzor [online]. [cit. 2021-03-29]. Dostupné z: https://www.alza.cz/time-of-flight-senzor
[31]RIISGAARD, Sren a BLAS, Morten Rufus. SLAM for Dummies [online]. [cit. 2021-04-20]. Dostupné z: https://dspace.mit. edu/bitstream/handle/1721.1/36832/16-412JSpring2004/NR/ rdonlyres/Aeronautics-and-Astronautics/16-412JSpring2004/ A3C5517F-C092-4554-AA43-232DC74609B3/0/1Aslam_blas_report.pdf
[32]MONTEMERLO, Michael a Sebastian THRUN. FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics [online]. Berlín: Springer-Verlag Berlin Heidelberg, 2007 [cit. 2021-04-20]. ISBN 978-3-540-46399-3.

[33]NAMOSHE, Molaletsa, Oduetse MATSEBE a Nkgatho TLALE. Feature extraction: techniques for landmark based navigation system. Sensor Fusion and its Applications [online]. Croatia: Sciyo, 2010, s. 347-373 [cit. 2021-04-20]. ISBN 978-953-307-101-5. Dostupné z: https://pdfs.semanticscholar.org/549a/ 8b14bcbf8b59e655480de2caa353da3ecaca.pdf
[34]An Iterative Closest Points Algorithm for Registration of 3D Laser Scanner Point Clouds with Geometric Features. Sensors [online]. 2017, , 3-7 [cit. 2021-04-20]. DOI: 10.3390/s17081862. Dostupné z: https://www.researchgate.net/publication/319072034_An_Iterative_ Closest_Points_Algorithm_for_Registration_of_3D_Laser_Scanner_ Point_Clouds_with_Geometric_Features
[35]CALONDER, Michael. EKF SLAM vs. FastSLAM - A Comparison [online]. Lausanne: Swiss Federal Institute of Technology, Computer Vision Lab [cit. 2021-04-20]. Dostupné z: https://infoscience.epfl.ch/record/146805/ files/ekf_fastslam_comp.pdf
[36]SU, Huaicheng. FastSLAM An Improved Particle FilteringAlgorithm for Simultaneous Localization andMapping: A Survey [online]. University of Windsor: School of Computer Science [cit. 2021-04-20]. Dostupné z: https://pdfs. semanticscholar.org/8e89/01197eeee430531c7661f236f06b5cd97bb1.pdf
[37]RPLIDAR. Https://www.slamtec.com/ [online]. 2018 [cit. 2021-4-10]. Dostupné z: http://bucket.download.slamtec.com/cd82fe93553fea5d15237cb3d6a45a406ef641aa/LR001 _SLAMTEC_rplidar_protocol_v2.0_en.pdf?fbclid=IwAR0QiSVmeSG6BPir3_BsueVFBnUD_Uup S6yaX0LB5AWwn1DuJUxvWL3RHAo

Seznam příloh

Adresář elektronické přílohy obsahuje:

A. JAW0013_Arduino

B. JAW0013_Processing

C. JAW0013_Python_